

"十二五"普通高等教育本科国家级规划教材 配套用书



21世纪大学本科计算机专业系列教材

蒋本珊 编著

计算机组成原理学习指导与习题解析（第3版）

http://www.tup.com.cn

- 根据教育部"高等学校计算机科学与技术专业规范"组织编写
- 与美国 ACM 和 IEEE CS *Computing Curricula* 最新进展同步
- 本书第2版被评为北京高等教育精品教材

清华大学出版社



"十二五"普通高等教育本科国家级规划教材 配套用书

21世纪大学本科计算机专业系列教材

计算机组成原理学习指导与习题解答

(第3版)

蒋本珊 编著

清华大学出版社

北京

内 容 简 介

本书是与“十二五”普通高等教育本科国家级规划教材《计算机组成原理(第 3 版)》完全配套的学习参考用书。全书共分 9 章,与主教材的章节完全相同,每一章都按基本内容摘要、重点难点梳理、典型例题详解和同步测试习题 4 个板块进行组织。

全书概念清楚,通俗易懂,由浅入深,意在通过典型例题的剖析,使读者能够加深对“计算机组成原理”课程所学知识的理解,熟练掌握单机系统范围内计算机的组织结构和基本工作原理,提高分析问题和解决问题的能力。

本书既是学习“计算机组成原理”课程时的重要参考书,又是报考计算机相关专业硕士研究生必不可少的考前复习资料。

图书在版编目(CIP)数据

计算机组成原理学习指导与习题解析/蒋本珊编著.—3 版.—北京:清华大学出版社,2014
21 世纪大学本科计算机专业系列教材
ISBN 978-7-302-36099-5

Ⅰ.①计…　Ⅱ.①蒋…　Ⅲ.①计算机组成原理—高等学校—教学参考资料　Ⅳ.①TP301

中国版本图书馆 CIP 数据核字(2014)第 069712 号

责任编辑:张瑞庆
封面设计:
责任校对:梁　毅
责任印制:

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦 A 座　　**邮　　编:**100084
社 总 机:010-62770175　　**邮　　购:**010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 刷 者:
装 订 者:
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张:**16.5　　**字　　数:**406 千字
版　　次:2014 年 6 月第 3 版　　**印　　次:**2014 年 6 月第1 次印刷
印　　数:1～　000
定　　价:　.00 元

产品编号:053792-01

本书主审：袁开榜

前言（第3版）

FOREWORD

《计算机组成原理学习指导与习题解析（第 2 版）》一书自 2009 年 8 月正式出版至今，已重印多次。2011 年，作为主教材《计算机组成原理（第 2 版）》的配套参考书，与主教材及另一本配套参考书《计算机组成原理教师用书（第 2 版）》一并被评为北京市精品教材。

2013 年 9 月，入选“十二五”普通高等教育本科国家级规划教材的《计算机组成原理（第 3 版）》正式出版，与主教材配套的辅助教材的修订工作也正式启动。本次修订主要的变化有：

（1）保留了原书的框架和风格，与主教材相同，增加了总线一章，使全书的总章数由 8 章变为 9 章；

（2）补充了部分重点难点梳理内容；

（3）针对全国硕士研究生入学统一考试计算机科学与技术学科联考计算机学科专业基础考试，在典型例题详解板块中增加了自 2009 年开始至 2014 年 6 年来计算机组成原理部分的全部真实考题，并进行了详细的解答和分析，为了与普通例题加以区别，真题在题号前用 * 号标注。

本书既是学生学习“计算机组成原理”课程时的指导和重要参考书，又是有志于报考计算机专业硕士研究生考生必不可少的考前复习资料。

本书在编写过程中，欧阳凌、潘海军帮助收集和整理了研究生入学统一考试的试题及部分例题和习题，并对各类题目及解答进行了审校，在此表示感谢。

作　者

2014 年 3 月于北京理工大学

前言（第2版）

FOREWORD

承蒙读者的厚爱，本书第1版出版至今三年多，已经多次重印。2007年，本书与主教材《计算机组成原理》和辅助教材《计算机组成原理教师用书》一起入选教育部普通高等教育“十一五”国家级规划教材。目前，《计算机组成原理》的相关教材已经形成了一个比较完整的教材教学体系，可以适应大多数高校的计算机及相关专业“计算机组成原理”课程教学的需要，受到了广大教师和学生的欢迎。

《计算机组成原理（第2版）》已于2008年9月正式出版，对辅助教材内容的更新也随之提上了议事日程，特别是2008年7月，教育部发布了“2009年全国硕士研究生入学统一考试计算机科学与技术学科联考计算机学科专业基础考试大纲”，计算机学科专业基础综合考试从2009年开始实行联合命题，统一考试。“计算机组成原理”课程作为主要的考试科目之一更是成为专业基础课中的重中之重。此次修订，保留了原书的框架和风格，全书章节保持不变，但与原书相比，进行了必要的调整，补充了大量新的例题、习题及其详细解答，涵盖了上述考试大纲的全部知识点。本书特针对研究生入学考试的新题型要求，大量增加了选择题，并对2009年1月的真实考题进行了详细的解答和分析，真题的题号前用*号标注。

希望本书不仅是学生学习“计算机组成原理”课程的指南和重要参考书，也是报考计算机专业硕士研究生必不可少的复习资料。

本书在编写过程中，欧阳凌帮助收集和整理了部分例题和习题，并对全部习题及解答进行了审校，在此表示感谢。

本书第1版自面市以来，收到了许多同行和读者发来的电子邮件，对于读者的来信本人均给予了回复和解答。希望修订之后的本书对您有所帮助，欢迎来信提出意见和建议。电子邮箱：bs.jiang@163.com。

作　者
2009年4月于北京理工大学

前言（第1版）

FOREWORD

“计算机组成原理”是计算机各类专业学生的必修核心课程之一，主要讨论计算机各大部件的基本组成原理，各大部件互连构成整机系统的技术。本课程在计算机学科中处于承上启下的地位，具有内容多、难度大等特点。根据读者学习“计算机组成原理”课程的需要，笔者参考、收集了与本课程有关的大量习题，最终通过整理编写成这本学习指导与习题解析。目的在于帮助读者更快地掌握计算机组成的基本原理和基本概念，学会使用科学的思维方式去分析并解决单机系统中的计算机组成的各种问题。

本书是与已列入中国计算机学会和清华大学出版社共同规划的“21 世纪大学本科计算机专业系列教材”之一的《计算机组成原理》一书完全配套的学习参考用书。全书共分 8 章，与主教材的章节完全相同，每一章都按基本内容摘要、重点难点梳理、典型例题详解和同步测试习题 4 个板块进行组织。

第一板块对基本的学习内容进行总结，列出了所涉及的主要知识点。

第二板块对重点与难点问题加以梳理，进行了比较详细的分析和讨论。

第三板块对典型的例题进行剖析，给出详尽的解答过程。

第四板块则给出各种类型的同步习题，供学生练习。习题后给出参考答案。

本书是根据中国计算机学会教育委员会制订的《中国计算机科学与技术学科教程2002（CCC2002）》对课程教学内容的要求，并结合作者多年从事本课程的教学经验编写而成的。全书力求做到概念清楚，通俗易懂，由浅入深，意在通过典型例题的剖析，使读者能够加深对“计算机组成原理”课程所学知识的理解，熟练掌握单机系统范围内计算机的组织结构和基本工作原理，提高分析问题和解决问题的能力。

本书是学好“计算机组成原理”课程的重要参考书，也可作为准备报考计算机相关专业硕士研究生考生的考前复习资料。

在本书编写过程中得到了“21 世纪大学本科计算机专业系列教材”编委会的多次指导和建议，清华大学出版社的编辑也为本书的出版做了许多工作。在此对他们辛勤的工作和热情的支持表示诚挚的感谢！

由于时间的原因以及个人的水平限制，书上难免出现错误和不妥之处，欢迎同行和广大读者批评指正。如有问题可直接与作者邮箱联系：bs.jiang@163.com。

作　者

2004 年 11 月于北京理工大学

目　录

CONTENTS

第 1 章 概 论

1.1 基本内容摘要

- 电子计算机与存储程序控制
  - 电子计算机的发展
  - 存储程序概念
- 计算机的硬件组成

  CPU＝运算器＋控制器；

  主机＝CPU＋主存储器；

  外部设备＝除去主机以外的硬件装置。
  - 计算机的主要部件

    输入设备、输出设备、存储器、运算器、控制器。
  - 各大部件之间连接

    计算机的总线结构；

    大、中型计算机的典型结构。
  - 不同对象观察到的计算机硬件系统

    一般用户观察到的计算机硬件系统；

    专业用户观察到的计算机硬件系统；

    计算机设计者观察到的计算机硬件系统。
  - 冯・诺依曼结构和哈佛结构的存储器设计思想

    冯・诺依曼结构；

    哈佛结构。
- 计算机系统
  - 硬件与软件的关系

    对于程序设计人员来说，硬件和软件在逻辑上是等价的。
  - 系列机和软件兼容
  - 计算机系统的多层次结构

    现代计算机系统是一个由硬件与软件组成的综合体，可以看成是按功能划分的多级层次结构。
  - 实际机器和虚拟机器

● 计算机的工作过程和主要性能指标
  ◆ 计算机的工作过程
  ◆ 计算机的主要性能指标

1.2 重点难点梳理

1. 存储程序概念

存储程序概念是冯·诺依曼等人首先提出来的,它可以简要地概括为以下几点:

(1) 计算机(指硬件)应由运算器、控制器、存储器、输入设备和输出设备五大基本部件组成;

(2) 计算机内部采用二进制来表示指令和数据;

(3) 将编好的程序和原始数据事先存入存储器中,然后再启动计算机工作。

存储程序概念中最重要的是第(3)点,通常把符合存储程序概念的计算机统称为冯·诺依曼型计算机。世界上第一台计算机 ENIAC 不是存储程序的计算机,它的存储容量极小,只能存储 20 个字长为 10 位的十进制数,程序不能事先存入存储器中。

2. 主机

中央处理器(CPU)和主存储器一起组成主机部分。

因为存储器有主存储器和辅助存储器之分,主机中只包括主存储器,而不包括辅助存储器。主存储器由 RAM 和 ROM 组成,对于微型计算机而言,是指插在主板上的内存条和其他存储芯片。辅助存储器则是硬盘、软盘、光盘等存储器的总称,它们处于主板之外,属于外部设备。

3. 总线

总线是一组能为多个部件服务的公共信息传送线路,它能分时地发送与接收各部件的信息。总线具有分时、共享的特点,即多个设备(或部件)挂在同一组总线上,但同一时刻只允许一个设备(或部件)发送信息。

最简单的总线结构是单总线结构。单总线并不意味着只有一根信号线,各大部件连接在单一的一组总线(系统总线)上,系统总线按传送信息的不同又可以细分为地址总线、数据总线和控制总线。地址总线由单方向的多根信号线组成,用于 CPU 向主存、外设传输地址信息;数据总线由双方向的多根信号线组成,CPU 可以沿这些线从主存或外设读入数据,也可以沿这些线向主存或外设送出数据;控制总线上传输的是控制信息,包括 CPU 送出的控制命令和主存(或外设)返回 CPU 的反馈信号。

4. 冯·诺依曼结构和哈佛结构的区别

冯·诺依曼结构和哈佛结构是指计算机中存储器的两种不同的设计思想,前者指令和数据是不加区别混合存储在同一个存储器中的,共享数据总线;后者指令和数据是完全分开的,存储器分为两部分,一个是程序存储器,用来存放指令,另一个是数据存储器,用来存放数据。

在冯·诺依曼结构中不能同时取指令和取操作数,而哈佛结构允许同时获取指令字(来自程序存储器)和操作数(来自数据存储器)。

5. 计算机系统

一个完整的计算机系统包含硬件系统和软件系统两大部分。

在计算机系统中没有一条明确的硬件与软件的分界线，硬件和软件之间的界面是浮动的。硬件软化可以增强系统的功能和适应性，软件硬化可以显著降低软件在时间上的开销。对于程序设计人员来说，硬件和软件在逻辑上是等价的。

6. 固件

固件是指那些存储在能永久保存信息的器件（如 ROM）中的程序，是具有软件功能的硬件。固件的性能指标介于硬件与软件之间，吸收了软件与硬件各自的优点，其执行速度快于软件，灵活性优于硬件，是软硬件结合的产物。

7. 系列机和软件兼容

系列机是指一个厂家生产的，具有相同的系统结构，但具有不同组成和实现的一系列不同型号的机器。

系列机具有软件兼容的特点，即同一个软件可以不加修改地运行于系统结构相同的各档机器上。软件兼容分为向上兼容、向下兼容、向前兼容和向后兼容 4 种。对系列机的软件向下和向前兼容可以不作要求，但必须保证向后兼容，力争做到向上兼容。

8. 实际机器和虚拟机器

现代计算机系统是一个由硬件与软件组成的综合体，可以看成是按功能划分的多级层次结构。实际机器是指由硬件或固件实现的机器，例如计算机系统的多层次结构中的硬件组成的实体、微程序级和传统机器级。虚拟机器是指以软件或以软件为主实现的机器，例如计算机系统的多层次结构中的操作系统级、汇编语言级、高级语言级和应用语言级。

虚拟机器只对该级的观察者存在，即在某一级观察者看来，他只需要通过该级的语言来了解和使用计算机，至于下级是如何工作和实现就不必关心了。

9. 计算机的主要性能指标

机器字长：指参与运算的数的基本位数，它是由加法器、寄存器的位数决定的，所以机器字长一般等于内部寄存器的大小。

数据通路宽度：数据总线一次所能并行传送信息的位数。这里所说的数据通路宽度实际是指外部数据总线的宽度。

主存容量：一个主存储器所能存储的全部信息量称为主存容量。对于字节编址的计算机，用字节数来表示主存容量，对于字编址的计算机，用字数乘以字长来表示主存容量。

运算速度：计算机的运算速度与许多因素有关。衡量运算速度的指标如下：

- 吞吐量　是指系统在单位时间内处理请求的数量。
- 响应时间　是指系统对请求作出响应的时间，响应时间包括 CPU 时间（运行一个程序所花费的时间）与等待时间（用于磁盘访问、存储器访问、I/O 操作、操作系统开销等时间）的总和。
- 主频　又称为时钟频率，表示在 CPU 内数字脉冲信号振荡的速度。
- CPU 时钟周期　主频的倒数就是 CPU 时钟周期，这是 CPU 中最小的时间元素。
- CPI　每条指令执行所用的时钟周期数。
- IPC　每个时钟周期执行的指令数（Instructions per Cycle）。在现代高性能计算机中，由于采用各种并行技术，使指令执行高度并行化，常常在一个时钟周期内可以执

行若干条指令。即

$$\mathrm{IPC}=\frac{1}{\mathrm{CPI}}$$

- CPU 执行时间　运行一个程序所花费的时间。

$$\text{CPU 执行时间}=\frac{\text{CPU 时钟周期数}}{\text{时钟频率}}=\frac{\mathrm{IC}\times\mathrm{CPI}}{\text{时钟频率}}$$

- MIPS　表示每秒执行多少百万条指令。MIPS 定义为

$$\mathrm{MIPS}=\frac{\text{指令条数}}{\text{执行时间}\times10^{6}}=\frac{\text{主频}}{\mathrm{CPI}}=\text{主频}\times\mathrm{IPC}$$

- MFLOPS　表示每秒执行多少百万次浮点运算。MFLOPS 定义为

$$\mathrm{MFLOPS}=\frac{\text{浮点操作次数}}{\text{执行时间}\times10^{6}}$$

随着计算机运算速度的不断提升,衡量运算速度的指标也在不断提升,出现了GFLOPS、TFLOPS、PFLOPS,它们之间的关系是:

一个 MFLOPS(Mega FLOPS)等于每秒 1 百万($=10^6$)次的浮点运算;

一个 GFLOPS(Giga FLOPS)等于每秒 10 亿($=10^9$)次的浮点运算;

一个 TFLOPS(Tera FLOPS)等于每秒 1 万亿($=10^{12}$)次的浮点运算;

一个 PFLOPS(Peta FLOPS)等于每秒 1 千万亿($=10^{15}$)次的浮点运算。

1.3　典型例题详解

【例 1.1】　冯·诺依曼计算机体系结构的基本思想是什么?按此思想设计的计算机硬件系统应由哪些部件组成?它们各起什么作用?

解:冯·诺依曼计算机体系结构的基本思想是存储程序概念,也就是将程序和数据一起存储在计算机中。计算机只要一启动,就能自动地取出一条条指令并执行之,直至程序执行完毕,得到计算结果为止。

按此思想设计的计算机硬件系统包括运算器、控制器、存储器、输入设备和输出设备五大基本部件。

运算器用来进行各种运算和数据转换;控制器则为计算机的工作提供统一的时钟和各种命令,协调计算机的各部件自动地工作;存储器用来存放程序、数据;输入设备和输出设备用来接收用户提供的外部信息或向用户提供输出信息。

【例 1.2】　如何理解软硬件之间的等价性?

解:计算机的大部分功能既能由硬件完成,也能由软件完成,从逻辑上讲两者是等效的。通常用硬件实现执行速度快,但成本高、修改困难,而用软件实现正相反。两者之间没有固定的界线。

【例 1.3】　微机 A 和 B 是采用不同主频的 CPU 芯片,片内逻辑电路完全相同。

(1) 若 A 机的 CPU 主频为 8MHz,B 机为 12MHz,则 A 机的 CPU 时钟周期为多少?

(2) 如 A 机的平均指令执行速度为 0.4MIPS,那么 A 机的平均指令周期为多少?

(3) B 机的平均指令执行速度为多少?

解:(1) A 机的 CPU 主频为 8MHz,所以 A 机的 CPU 时钟周期 $=1\div8\mathrm{MHz}=$

0.125μs。

（2）A 机的平均指令执行速度为 0.4MIPS，所以 A 机的平均指令周期＝1÷0.4MIPS＝2.5μs。

（3）A 机平均每条指令的时钟周期数＝2.5μs÷0.125μs＝20。而微机 A 和 B 片内逻辑电路完全相同，所以 B 机平均每条指令的时钟周期数也为 20。

由于 B 机的 CPU 主频为 12MHz，所以 B 机的 CPU 时钟周期＝1÷12MHz＝$\frac{1}{12}$μs。

B 机的平均指令周期＝20×$\frac{1}{12}$μs＝$\frac{5}{3}$μs。

B 机的平均指令执行速度＝$\frac{3}{5}$MIPS＝0.6MIPS。

【例 1.4】 计算 Pentium Ⅱ 450 处理机的运算速度。

解：由于 Pentium Ⅱ 450 处理机的 IPC＝2（或 CPI＝0.5），主频＝450MHz，因此，

$$\text{MIPS}_{\text{Pentium II 450}}=\text{主频}\times\text{IPC}=450\times 2=900\text{MIPS}$$

*【例 1.5】 下列选项中，能缩短程序执行时间的措施是________。

Ⅰ. 提高 CPU 时钟频率　　Ⅱ. 优化数据通路结构　　Ⅲ. 对程序进行编译优化

A. 仅Ⅰ和Ⅱ　　B. 仅Ⅰ和Ⅲ　　C. 仅Ⅱ和Ⅲ　　D. Ⅰ、Ⅱ和Ⅲ

解：D。

分析：一般来讲，CPU 时钟频率（主频）越高，CPU 的速度就越快；优化数据通路结构，可以有效提高计算机系统的吞吐量；编译优化可得到更优的指令序列。所以Ⅰ、Ⅱ、Ⅲ都是缩短程序执行时间的措施。

*【例 1.6】 下列选项中，描述浮点数操作速度指标的是________。

A. MIPS　　B. CPI　　C. IPC　　D. MFLOPS

解：D。

分析：MFLOPS 表示每秒执行多少百万次浮点运算，用来描述计算机的浮点运算速度，适用于衡量向量机的性能。

*【例 1.7】 假定基准程序 A 在某计算机上的运行时间为 100 秒，其中 90 秒为 CPU 时间，其余为 I/O 时间。若 CPU 速度提高 50%，I/O 速度不变，则运行基准程序 A 所耗费的时间是________。

A. 55 秒　　B. 60 秒　　C. 65 秒　　D. 70 秒

解：D。

分析：CPU 速度提高 50%，即 CPU 性能提高比为 1.5，改进之后的 CPU 运行时间＝90÷1.5＝60 秒。I/O 速度不变，仍维持 10 秒，所以运行基准程序 A 所耗费的时间为 70 秒。

*【例 1.8】 某计算机主频为 1.2GHz，其指令分为 4 类，它们在基准程序中所占比例及 CPI 如表 1-1 所示。

该机的 MIPS 数是________。

A. 100　　B. 200　　C. 400　　D. 600

解：C。

分析：首先根据 4 类指令在基准程序中所占比例及 CPI，可得出：

表 1-1 各类指令在基准程序中所占比例及 CPI

指令类型	所占比例	CPI
A	50%	2
B	20%	3
C	10%	4
D	20%	5

平均 $CPI=0.5\times2+0.2\times3+0.1\times4+0.2\times5=3$。

已知计算机主频为 1.2GHz,所以 $MIPS=\dfrac{主频}{CPI}=1200MHz\div3=400$。

*【例 1.9】 程序 P 在机器 M 上的执行时间是 20 秒,编译优化后,P 执行的指令数减少到原来的 70%,而 CPI 增加到原来的 1.2 倍,则 P 在 M 上的执行时间是________。

A. 8.4 秒　　B. 11.7 秒　　C. 14.0 秒　　D. 16.8 秒

解:D。

分析:$CPU执行时间=\dfrac{CPU时钟周期数}{时钟频率}=\dfrac{IC\times CPI}{时钟频率}$

由于机器 M 的时钟频率不变,在编译优化后,$IC_{新}=0.7\times IC_{旧}$,$CPI_{新}=1.2\times CPI_{旧}$,

$CPU执行时间_{新}=CPU执行时间_{旧}\times(IC_{新}\div IC_{旧})\times(CPI_{新}\div CPI_{旧})=20\times0.7\times1.2=16.8$ 秒。

1.4 同步测试习题及解答

1.4.1 同步测试习题

一、填空题

1. 冯·诺依曼结构的特点是________。
2. 主机由 CPU 和________组成。
3. 现在主要采用________结构作为微/小型计算机硬件之间的连接方式。
4. 计算机系统由________系统和________系统构成。
5. 计算机系统的多层次结构中,位于硬件之外的所有层次统称为________。

二、选择题

1. 通常划分计算机发展时代是以________为标准的。

A. 所用电子器件　　B. 运算速度　　C. 计算机结构　　D. 所有语言

2. 电子计算机技术在 60 多年中虽有很大的进步,但至今其运行仍遵循着一位科学家提出的基本原理。这位科学家是________。

A. 牛顿　　B. 爱因斯坦　　C. 爱迪生　　D. 冯·诺依曼

3. 冯·诺依曼计算机结构的核心思想是________。

A. 二进制运算　　B. 有存储信息的功能

C. 运算速度快　　D. 存储程序控制

4. 电子计算机可分为数字计算机、模拟计算机和数模混合计算机，它是按照________。
   A. 计算机的用途分类　　B. 计算机的使用方式分类
   C. 信息的形式和处理方式分类　　D. 计算机的系统规模分类
5. 完整的计算机系统应包括________。
   A. 运算器、存储器、控制器　　B. 外部设备和主机
   C. 主机和实用程序　　D. 配套的硬件设备和软件系统
6. 中央处理器(CPU)是指________。
   A. 运算器　　B. 控制器
   C. 运算器和控制器　　D. 运算器和存储器
7. 计算机的存储器系统是指________。
   A. RAM　　B. ROM
   C. 主存储器　　D. Cache、主存储器和辅助存储器
8. 目前我们所说的个人计算机属于________。
   A. 巨型机　　B. 中型机　　C. 小型机　　D. 微型机
9. 微型计算机的发展以________技术为标志。
   A. 操作系统　　B. 微处理器　　C. 磁盘　　D. 软件
10. 对计算机的软、硬件资源进行管理，是________的功能。
   A. 操作系统　　B. 数据库管理系统
   C. 语言处理程序　　D. 用户程序
11. 以下软件中________是计算机系统软件。
   A. 数据处理软件　　B. 操作系统软件，语言编译软件
   C. 办公自动化软件　　D. Word 软件
12. 计算机硬件能够直接执行的只有________。
   A. 机器语言　　B. 汇编语言
   C. 机器语言和汇编语言　　D. 各种高级语言
13. 只有当程序执行时，它才会去将源程序翻译成机器语言，而且一次只能读取、翻译并执行源程序中的一行语句，此程序称为________。
   A. 目标程序　　B. 编译程序　　C. 解释程序　　D. 汇编程序
14. 用于科学计算的计算机中，标志系统性能的主要参数是________。
   A. 主频　　B. 主存容量　　C. MIPS　　D. MFLOPS

三、判断题

1. 存储程序的基本含义是将编制好的程序和原始数据事先存入主存储器中。（　　）
2. 利用大规模集成电路技术把计算机的运算部件和控制部件做在一块集成电路芯片上，这样的一块芯片叫做单片机。（　　）
3. 计算机“运算速度”指标的含义是指每秒钟能执行多少条操作系统的命令。（　　）

1.4.2　同步测试习题解答

一、填空题

1. 存储程序。

2. 主存储器。
3. 总线。
4. 硬件,软件。
5. 虚拟机器。

二、选择题

1. A。通常按计算机所采用的微电子器件的发展将计算机分代。
2. D。冯·诺依曼提出了“存储程序”的概念。
3. D。通常把符合存储程序概念的计算机统称为冯·诺依曼型计算机。
4. C。根据计算机的用途可分为通用计算机和专用计算机;根据计算机系统的规模可分为巨型机、大型机、中型机、小型机、微型机等;根据信息的形式和处理方式可分为电子数字计算机、电子模拟计算机,也可以有数模混合计算机。
5. D。一个完整的计算机系统应包括硬件系统和软件系统两大部分,硬件和软件是相辅相成的,不可分割的整体。
6. C。CPU 包括运算器和控制器。
7. D。三级存储系统包括 Cache、主存储器和辅助存储器。
8. D。个人计算机是指微型计算机。
9. B。微型计算机的发展是以微处理器的发展为标志的。
10. A。操作系统的任务是对计算机的软、硬件资源进行管理。
11. B。操作系统和语言处理程序都属于系统软件的范畴。
12. A。机器语言是计算机唯一可以直接识别和执行的语言。
13. C。将高级语言编写的源程序翻译成机器语言的语言处理程序包括编译程序和解释程序,前者是先将源程序转换为目标程序,再开始执行;而后者对源程序的处理是采用一行一行语句边解释边执行的方法。
14. D。MFLOPS 表示每秒钟执行百万次浮点运算,用来描述计算机的浮点运算速度,而科学计算的计算机更看重浮点运算速度。

三、判断题

1. √。
2. ×。这样的芯片应称为 CPU。
3. ×。应指每秒执行多少条指令或每秒执行多少次浮点运算。

第 2 章 数据的机器层次表示

2.1 基本内容摘要

- 数值数据的表示
  - 计算机中的数值数据

    不同数制的表示。
  - 无符号数和带符号数的区别
  - 原码表示法

    原码真值 0 的两种不同的表示形式。
  - 补码表示法

    补码真值 0 的唯一的表示形式。
  - 反码表示法

    反码真值 0 的两种不同的表示形式。
  - 3 种机器数的比较与转换
- 机器数的定点表示与浮点表示
  - 定点表示法

    定点小数的表示范围；

    定点整数的表示范围。
  - 浮点表示法

    浮点数的表示范围；

    规格化的浮点数。
  - 浮点数阶码的移码表示法
  - IEEE 754 标准浮点数
  - 定点/浮点表示法与定点/浮点计算机
- 非数值数据的表示
  - 字符和字符串的表示

    ASCII 字符编码；

    字符串的存放。
  - 汉字的表示

    汉字国标码；

汉字区位码；
汉字机内码；
汉字字形码。
- 统一代码

- 十进制数和数串的表示
  - 十进制数的编码(二-十进制编码)
    8421 码；
    2421 码；
    余 3 码；
    Gray 码。
  - 十进制数串
    非压缩的十进制数串；
    压缩的十进制数串。
- 不同类型的数据表示举例
  - C 语言中的数据表示
  - 现代微型计算机系统中的数据表示
- 数据校验码
  - 奇偶校验码
    奇偶校验概念；
    简单奇偶校验与交叉奇偶校验。
  - 汉明校验码
  - 循环冗余校验码

2.2 重点难点梳理

1. 无符号数与带符号数的区别

无符号数就是整个机器字长的全部二进制位均表示数值位(没有符号位)，相当于数的绝对值。

带符号数在日常生活中用＋、－号加绝对值来表示数值的大小，由于计算机无法识别＋、－号，所以在计算机中需要将符号数码化。通常，约定二进制数的最高位为符号位，0 表示正号，1 表示负号。带符号数在计算机中使用的表示数的形式称为机器数，常见的带符号机器数有原码、反码、补码等不同的表示形式。

数据由带符号数转换为同一长度的无符号数时，原来的符号位不再是符号位，而成为数据的一部分，所以负数转换成无符号数时，数值将发生改变。数据由无符号数转换为同一长度的带符号数时，各个二进制位的状态不变，但最高位被当作符号位，这时也会发生数值改变。

2. 不同机器数中真值 0 的表示方法

对于真值 0，原码和反码各有两种不同的表示形式，而补码只有唯一的一种表示形式。

假设字长为 8 位，则

$$[+0]_{原}=00000000$$

$$[-0]_{原}=10000000$$

$$[+0]_{补}=[-0]_{补}=00000000$$

$$[+0]_{反}=00000000$$

$$[-0]_{反}=11111111$$

3. 3种机器数的主要区别

原码、补码和反码的区别有以下几点：

(1) 对于正数它们都等于真值本身，而对于负数各有不同的表示。

(2) 最高位都表示符号位，补码和反码的符号位可作为数值位的一部分看待，和数值位一起参加运算，但原码的符号位不允许和数值位同等看待，必须分开进行处理。

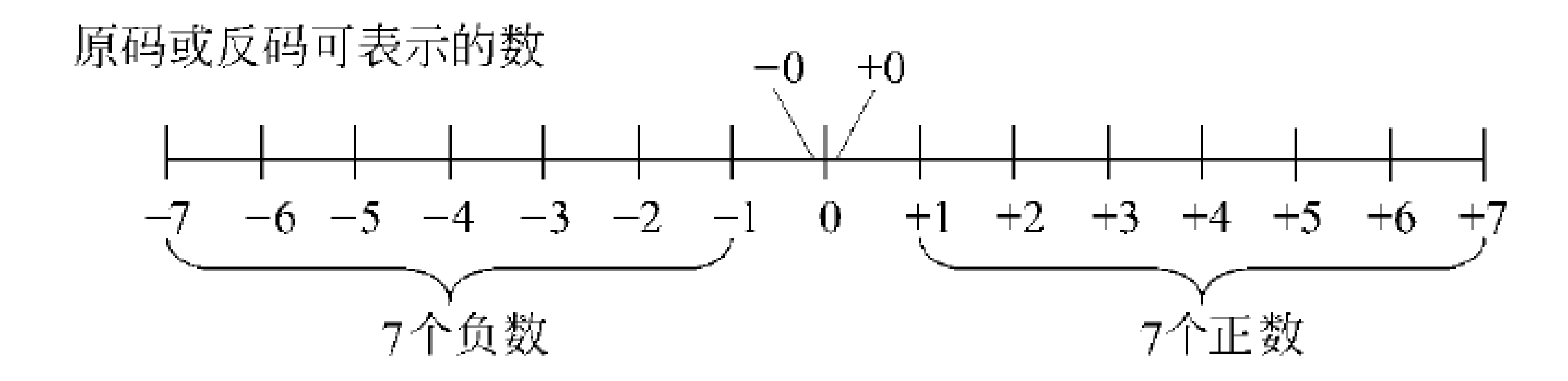
(3) 对于真值0，原码和反码各有两种不同的表示形式，而补码只有唯一的一种表示形式。

(4) 原码、反码表示的正、负数范围相对零来说是对称的，但补码负数表示范围较正数表示范围宽，能多表示一个最负的数(绝对值最大的负数)。

4. 补码表示范围比原码宽

这个问题是与真值0的问题密切相关的。因为原码和反码的真值0各有两种不同的表示形式，而补码只有唯一的一种表示形式。

以字长为4位的二进制整数为例，一共有 2^4 种不同的代码，对于原码来说，因为有+0和−0两个不同的编码，所以总共可以表示7个正整数和7个负整数，正、负数范围相对零来说是对称的。而补码的+0和−0表示形式相同，这样就多出来一个代码(1000)。这个代码所对应的真值是−8，所以补码总共可以表示7个正整数和8个负整数，负数表示范围较正数表示范围宽，能多表示一个最负的数(绝对值最大的负数)。原码、补码、反码可表示的数如图2-1所示。



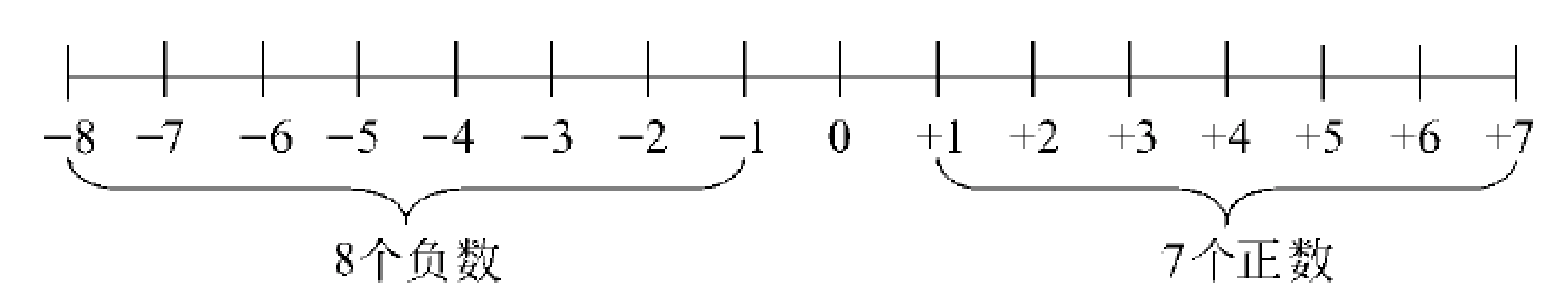


图2-1　原码、补码、反码可表示的数

表2-1给出了二进制代码与3种机器数对应十进制真值的关系，其中代码1000是需要特别关注的。当这个代码是原码时，对应的真值为−0；当这个代码是反码时，对应的真值是−7。而当这个代码是补码时，对应的真值为−8，此时可以认为最高位的“1”有两个含义，既代表负号，又代表这一位的位权 $2^3=8$。这个数在数轴上处于最左边，称为绝对值最大的负数，也可称为最小负数。这个绝对值最大的负数，对于字长为 $n+1$ 位的定点整数，其值等于

-2^n(如字长为8位,其值等于$-2^7=-128$);对于定点小数,其值等于$-2^0=-1$。

表 2-1　二进制代码与3种机器数对应十进制真值的关系

二进制代码	对应的十进制真值			二进制代码	对应的十进制真值		
	原码	补码	反码		原码	补码	反码
0000	0	0	0	**1000**	−0	−8	−7
0001	1	1	1	1001	−1	−7	−6
0010	2	2	2	1010	−2	−6	−5
0011	3	3	3	1011	−3	−5	−4
0100	4	4	4	1100	−4	−4	−3
0101	5	5	5	1101	−5	−3	−2
0110	6	6	6	1110	−6	−2	−1
0111	7	7	7	1111	−7	−1	−0

5. 定点数的表示范围

在定点表示法中,参加运算的数以及运算的结果都必须保证落在该定点数所能表示的数值范围内,定点数的表示范围如图2-2所示。我们最关注的3个值分别是最大正数、最小正数和绝对值最大负数(也称为最小负数)。

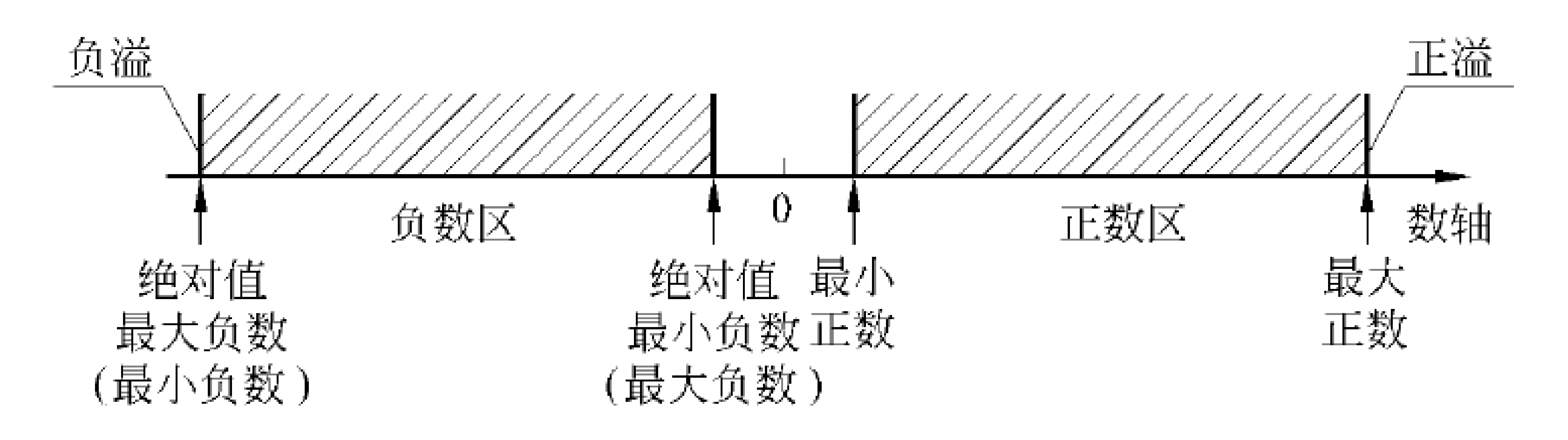


图 2-2　定点数的表示范围

定点小数是一个纯小数,小数点位置固定在最高有效数位之前,符号位之后。设机器字长有$n+1$位,记作$X_s.X_1X_2\cdots X_n$。

$$X_{最大正数}=(1-2^{-n})$$

$$X_{最小正数}=2^{-n}$$

当X为负数时,情况要稍微复杂一些,这是因为在计算机中带符号数可用补码表示,也可用原码表示,原码和补码的表示范围有一些差别。

若机器数为原码表示,则

$$X_{绝对值最大负数}=-(1-2^{-n})$$

所以原码定点小数表示范围为$-(1-2^{-n})\sim(1-2^{-n})$。例如,某机字长为16位,定点小数的表示范围为:

$$-(1-2^{-15})\sim+(1-2^{-15})$$

即

$$\underbrace{1.111\cdots1}_{16位个1}\sim0.\underbrace{111\cdots1}_{15位个1}$$

若机器数为补码表示,则

$$X_{绝对值最大负数}=-1$$

所以补码定点小数表示范围为$-1\sim(1-2^{-n})$。例如，某机字长为16位，定点小数的表示范围为：

$$-1\sim+(1-2^{-15})$$

即

$$1.\underbrace{000\cdots0}_{15\text{位个}0}\sim 0.\underbrace{111\cdots1}_{15\text{位个}1}$$

定点整数是一个纯整数，小数点位置隐含固定在最低有效数位之后。设机器字长有$n+1$位，记作$X_sX_1X_2\cdots X_n$。

根据前述方法不难推出：

$$X_{\text{最大正数}}=(2^n-1)$$

$$X_{\text{最小正数}}=1$$

$$X_{\text{绝对值最大负数}}=-(2^n-1)\text{（原码表示时）}$$

$$X_{\text{绝对值最大负数}}=-2^n\text{（补码表示时）}$$

综上所述，原码定点整数的表示范围为$-(2^n-1)\sim(2^n-1)$。例如，某机字长为16位，定点整数的表示范围为：

$$-(2^{15}-1)\sim+(2^{15}-1)$$

即

$$\underbrace{1111\cdots1}_{16\text{位个}1}\sim 0\underbrace{111\cdots1}_{15\text{位个}1}$$

补码定点整数的表示范围为$-2^n\sim(2^n-1)$。例如，某机字长为16位，定点整数的表示范围为：

$$-2^{15}\sim+(2^{15}-1)$$

即

$$1\underbrace{000\cdots0}_{15\text{位个}0}\sim 0\underbrace{111\cdots1}_{15\text{位个}1}$$

6. 浮点数的格式和表示范围

浮点数N表示为

$$N=M\times r^E$$

其中，r是浮点数阶码的底，也称为尾数基数，通常$r=2$。E（阶码部分）和M（尾数部分）都是带符号的定点数，在大多数计算机中，尾数为纯小数，常用原码或补码表示；阶码为纯整数，常用移码或补码表示。

浮点数的格式可以有多种形式，由计算机设计者决定，在计算机表示、存储和处理浮点数据时都遵守该格式即可。浮点数的一般格式如图2-3所示，k和n分别表示阶码和尾数的位数（不包括符号位）。

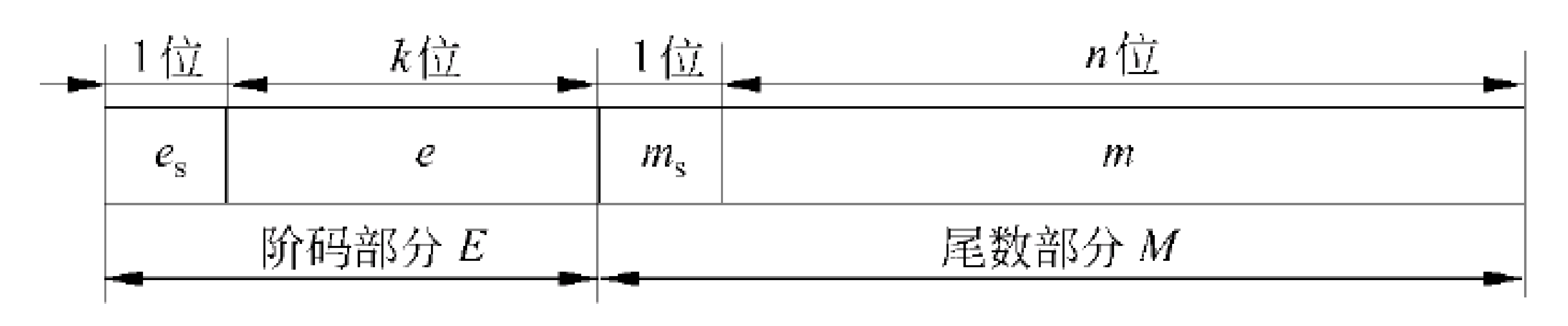


图 2-3　浮点数的一般格式

假设尾数和阶码均用补码表示。当 $e_s=0$,$m_s=0$,阶码和尾数的数值位各位全为1(即阶码和尾数都为最大正数)时,该浮点数为最大正数:

$$X_{最大正数}=(1-2^{-n})\times 2^{2^k-1}$$

当 $e_s=1$,$m_s=0$,尾数的最低位 $m_n=1$,其余各位为0(即阶码为绝对值最大的负数,尾数为最小正数)时,该浮点数为最小正数:

$$X_{最小正数}=2^{-n}\times 2^{-2^k}$$

当 $e_s=0$,阶码的数值位为全1;$m_s=1$,尾数的数值位为全0(即阶码为最大正数,尾数为绝对值最大的负数)时,该浮点数为绝对值最大负数:

$$X_{绝对值最大负数}=-1\times 2^{2^k-1}$$

图2-4所示的另一种浮点数格式在计算机中使用得非常普遍,这是因为此时将尾数的符号位放在最高位(MSB)的位置上,与定点数相一致,便于判定数的正负。此时假设 E(阶码部分)共 $k+1$ 位,数符1位,尾数数值 m 位。由于尾数部分用原码表示,所以最大正数和最小正数的值没有变化,绝对值最大的负数值为:

$$X_{绝对值最大负数}=-(1-2^{-n})\times 2^{2^k-1}$$

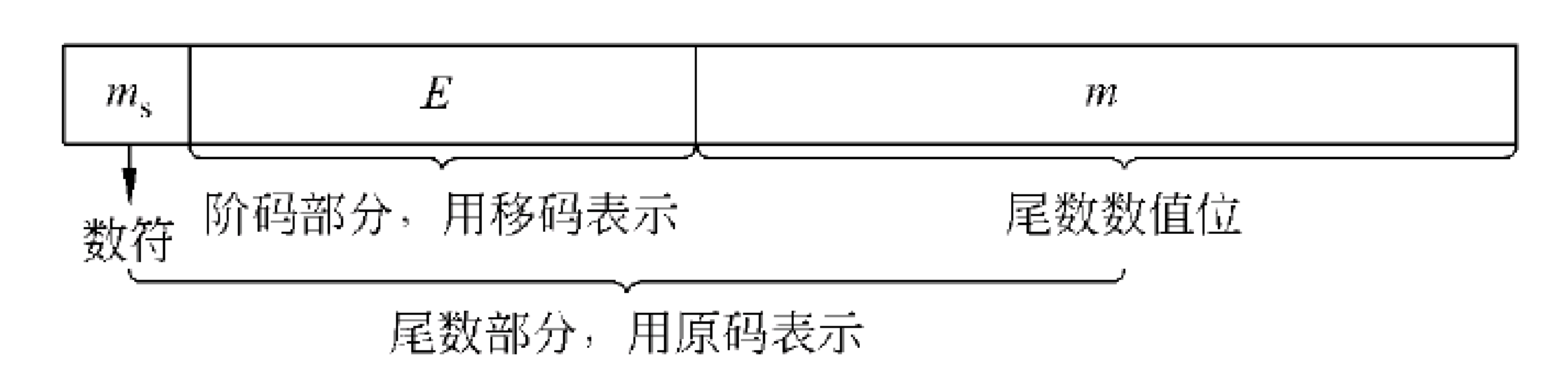


图2-4 另一种浮点数格式

7. 规格化浮点数

为了提高运算的精度,通常采取浮点数规格化形式,即规定尾数的最高数位必须是一个有效值。

通常尾数的基数为2时,规格化表示的尾数形式如下。

正数:0.1××…×。其最大值表示为0.111…1,最小值表示为0.100…0。尾数的表示范围 $1/2\leqslant M<1$。

负数(原码):1.1××…×。其最大值表示为1.100…0,最小值表示为1.111…1。尾数的表示范围 $-1<M\leqslant -1/2$。

负数(补码):1.0××…×。其最大值表示为1.011…1,最小值表示为1.000…0。尾数的表示范围 $-1\leqslant M<-1/2$。

在图2-3格式下,若尾数和阶码均用补码表示,当 $e_s=1$,阶码的数值部分均为0;$m_s=0$,尾数的最高位 $m_1=1$,其余各位为0时,该浮点数为规格化的最小正数:

$$X_{规格化的最小正数}=2^{-1}\times 2^{-2^k}$$

当 $e_s=1$,阶码的数值部分均为0;$m_s=1$,尾数的最高位 $m_1=0$,其余各位为1时,该浮点数为规格化的绝对值最小负数:

$$X_{规格化的绝对值最小负数}=-(2^{-1}+2^{-n})\times 2^{-2^k}$$

而在图2-4格式下,规格化的绝对值最小负数和规格化的最小正数数值相同,只是相差一个符号而已。

显然,规格化浮点数的表示范围要小于非规格化浮点数的表示范围。

表 2-2 列出了图 2-3 格式下浮点数的几个典型值，此时阶码和尾数均用补码表示，阶码部分共 $k+1$ 位（含一位阶符），尾数部分共 $n+1$ 位（含一位尾符）。如果采用图 2-4 的格式，这些典型值的阶码和尾数的代码以及真值可能会有所变化，具体的变化留待读者思考。

表 2-2　浮点数的典型值

浮点数的典型值	浮点数代码		真　值
	阶　码	尾　数	
最大正数	01…1	0.11…11	$(1-2^{-n})\times 2^{2^k-1}$
绝对值最大负数	01…1	1.00…00	$-1\times 2^{2^k-1}$
最小正数	10…0	0.00…01	$2^{-n}\times 2^{-2^k}$
规格化的最小正数	10…0	0.10…00	$2^{-1}\times 2^{-2^k}$
绝对值最小负数	10…0	1.11…11	$-2^{-n}\times 2^{-2^k}$
规格化的绝对值最小负数	10…0	1.01…11	$-(2^{-1}+2^{-n})\times 2^{-2^k}$

应当提醒引起注意的是，在解题过程中，一定要首先看清楚浮点数的格式以及阶码、尾数部分采用何种机器数，因为格式和机器数的不同会使结果有一些不同。

8. 浮点数阶码的移码表示法

浮点数的阶码是带符号的定点整数，在多数通用计算机中，采用移码表示。

移码就是在真值 X 基础上加一个常数，这个常数被称为偏置值，即

$$[X]_{移} = 偏置值 + X$$

移码把真值映射到一个正数域，所以可将移码视为无符号数。对于字长为 $n+1$ 位的定点整数而言，偏置值通常为 2^n。

根据补码的定义，有：

$$[X]_{补} = 2^{n+1} + X = 2^n + 2^n + X = 2^n + [X]_{移}$$

从上面的推导结果可以看出，同一数值的移码和补码除最高位（MSB）相反外，其他各位相同。假设字长为 8 位，则有：

$$[+0]_{补} = [-0]_{补} = 00000000$$
$$[+0]_{移} = [-0]_{移} = 10000000$$

阶码之所以采用移码表示最主要的原因在于：

（1）便于比较浮点数的大小。阶码小的，对应的真值就小；阶码大的，其对应的真值就大。如阶码有 8 位，移码为全 0 时，表示真值最小，为 -128；移码为全 1 时，表示真值最大，为 $+127$。移码的大小直观反映了真值的大小，不必考虑符号问题，这使得浮点运算中的阶码比较很方便。

（2）简化机器中的判零电路。当阶码全为 0，尾数也全为 0 时，表示机器零。对于浮点数 $N=M\times r^E$，当尾数 $M=0$ 时，不论其阶码为何值都有 $N=0$。另一种情况，当 $E<-2^n$ 时，$M\neq 0$，此时 $N\neq 0$ 但非常接近于 0，一般以 $N=0$ 处理。为了保证唯一性，要求规定一个标准的浮点数零的表示形式，称为“机器零”，它应该同时具有 0 的尾数和最小阶码（全 0）。

9. IEEE 754 标准的浮点数

IEEE 754 标准浮点数的格式与图 2-4 所示浮点数格式相同，唯一的区别在于它采用隐

含尾数最高数位的方法，这样，无形中又增加了一位尾数。

规格化浮点数隐含的最高数位1是一位整数(即位权为 2^0)，即相当于尾数扩大了一倍(左移了1位)。为保持该浮点数的值不变，阶码就应当相应的减1，因此，短浮点数(32位)格式中偏置值取127(128－1)。在长浮点数(64位)格式中偏置值取1023(1024－1)，这就是为什么IEEE 754标准的短浮点数阶码的偏置值为127，长浮点数阶码的偏置值为1023的原因。

IEEE 754中，规格化的短浮点数 v 的真值表示为：

$$v = (-1)^S \times (1.f) \times 2^{E-127}$$

规格化的长浮点数 v 的真值表示为：

$$v = (-1)^S \times (1.f) \times 2^{E-1023}$$

IEEE 754浮点数的解释见表2-3。

表2-3　IEEE 754浮点数的解释

名　称	短浮点数(32位)				长浮点数(64位)			
	符号	移码阶码	尾数	值	符号	移码阶码	尾数	值
正零	0	0	0	0	0	0	0	0
负零	1	0	0	－0	1	0	0	－0
正无穷大	0	255(全1)	0	∞	0	2047(全1)	0	∞
负无穷大	1	255(全1)	0	$-\infty$	1	2047(全1)	0	$-\infty$
非数(N_aN)	0或1	255(全1)	$\neq 0$	N_aN	0或1	2047(全1)	$\neq 0$	N_aN
正规格化非零数	0	$0<E<255$	f	$2^{E-127}(1.f)$	0	$0<E<2047$	f	$2^{E-1023}(1.f)$
负规格化非零数	1	$0<E<255$	f	$-2^{E-127}(1.f)$	1	$0<E<2047$	f	$-2^{E-1023}(1.f)$
正非规格化	0	0	$f\neq 0$	$2^{E-126}(0.f)$	0	0	$f\neq 0$	$2^{E-1022}(0.f)$
负非规格化	1	0	$f\neq 0$	$-2^{E-126}(0.f)$	1	0	$f\neq 0$	$-2^{E-1022}(0.f)$

下面以32位的短浮点数为例说明(E 表示移码阶码，f 表示尾数)。

1) 零

零有两种表示：＋0和－0。

$E=0$，$f=0$，尾数的隐含位为0，所表示的浮点数为0(机器零)。

2) 无穷大数

$E=255$，$f=0$，所表示的浮点数为无穷大。

3) 非数 N_aN

$E=255$，$f\neq 0$，表示一个非数值。N_aN 用来通知各种例外条件，可以使计算在出现异常时能够继续进行下去。没有数学解释的操作(例如0除以0)将产生一个非数值 N_aN。

4) 规格化数

此时 $1\leqslant E\leqslant 254$，IEEE 754采用隐含尾数最高数位1的方法，因此尾数实际上是24位(1位隐含位＋23位小数位)。规格化浮点数的尾数为 $1.f$，所表示的规格化浮点数为 $\pm 2^{E-127}\times(1.f)$。

5) 非规格化数

此时 $E=0$，$f\neq 0$，尾数的隐含位为0，所表示的非规格化浮点数为 $\pm 2^{E-126}\times(0.f)$。非规格化数用于表示某些下溢数据，它的大小在零与最小有限数之间。

10. ASCII 码的编码规律

在 ASCII 码中，数字和英文字母都是按顺序排列的，只要知道其中一个数字或英文字母的二进制代码，不要查表就可以推导出其他数字或字母的二进制代码。

ASCII 码中 0～9 十个数字的编码为 011××××，其中最后 4 位××××恰恰是 0000～1001，正好与它们的二进制编码相同，这不但使十进制数字进入计算机后易于压缩成 4 位代码，而且也便于进一步的机内信息处理。

ASCII 码中英文字母无论大小写都满足特定的规律，如 A 是大写的第 1 个字母，其 ASCII 码为 41H(1000001)，则字母 Z(第 26 个字母)的 ASCII 码为 5AH(1011010)，即 40H＋26(1AH)＝5AH。

11. 3 种汉字编码的区别

1) 汉字国标码

汉字国标码又称为汉字交换码，主要用于汉字信息处理系统之间或者通信系统之间交换信息使用。如 GB 2312—80 中规定每个汉字、图形符号都用两个字节表示(GB 码)。

2) 汉字区位码

这是一种输入码，区位码定长 4 位，前 2 位表示区号，后 2 位表示位号，汉字的区号和位号均用十进制数表示。它将 GB 2312—80 中的汉字分为 94 个区，每个区中包含 94 个汉字(位)，区和位组成一个二维数组，每个汉字在数组中对应一个唯一的区位码。

3) 汉字机内码

这是汉字在计算机内部的编码，GB 码的机内码也是两字节长的代码，它是在相应 GB 码的每个字节最高位上加“1”，以免在当系统中同时存在 ASCII 码和汉字国标码时产生二义性。

前 3 种汉字编码的关系：

汉字国标码 ＝ 汉字区位码(十六进制)＋2020H

汉字机内码 ＝ 汉字国标码＋8080H

汉字机内码 ＝ 汉字区位码(十六进制)＋A0A0H

通常，汉字的国标码和机内码都用十六进制数表示，而汉字区位码用十进制数表示，所以在 3 种汉字编码的转换时，千万不要忘记先将十进制的区位码变成十六进制之后，再利用上述关系式进行转换。

除了上述 3 种汉字编码以外，还有一种汉字编码——汉字字形码，这是一种输出码，汉字字形码的字节数与汉字的输出质量有关。

12. 十进制数的 BCD 编码

4 位二进制数可以组合出 16 种代码，其中 10 种代码表示 0～9 这 10 个数码，而其余的 6 种代码就是非法码。由于可以取任意的 10 种代码来表示 10 个数码，所以就可能产生多种 BCD 编码。BCD 编码既具有二进制数的形式，又保持了十进制数的特点，可以作为人机联系的一种中间表示，也可以用它直接进行运算。表 2-4 列出了几种常见的 BCD 编码。

表 2-4　常见的 BCD 编码

码的类型		8421 码		2421 码		余 3 码	
0	0000	0	0000	0	0000	非法码	
1	0001	1	0001	1	0001		
2	0010	2	0010	2	0010		
3	0011	3	0011	3	0011	0	0011
4	0100	4	0100	4	0100	1	0100
5	0101	5	0101	非法码		2	0101
6	0110	6	0110			3	0110
7	0111	7	0111			4	0111
8	1000	8	1000			5	1000
9	1001	9	1001			6	1001
10	1010	非法码				7	1010
11	1011			5	1011	8	1011
12	1100			6	1100	9	1100
13	1101			7	1101	非法码	
14	1110			8	1110		
15	1111			9	1111		

BCD 码用 4 位二进制数来表示 1 位十进制数,如十进制数 3609 可以分别表示为:

$$(3609)_{10} = (0011\ 0110\ 0000\ 1001)_{8421码}$$
$$= (0011\ 1100\ 0000\ 1111)_{2421码}$$
$$= (0110\ 1001\ 0011\ 1100)_{余3码}$$

注意:有人把 8421 码与 BCD 码混为一谈,这是不准确的,8421 码只是 BCD 码中的一种而已。

十进制数的 Gray 码也是 BCD 码,它与前几种编码不同的是它属于可靠性编码,是一种错误最小化的编码方式,它的编码规则是使相邻两代码之间只有一个二进制位的状态不同,其余 3 个二进制位必须有相同状态,并且具有封闭循环性,所以十进制数的 Gray 码有很多种。十进制数的 Gray 码是无权码,每一个二进制位没有确定的大小,不能直接进行比较大小和算术运算。

13. 奇偶校验码与奇偶校验位

奇偶校验码是一种最简便、最直观、应用最广泛的检错码,它只能检出一位错或奇数位错,但无法给错误定位,因此不能纠正错误,所以常用于对存储器数据的检查或者传输数据的检查。

奇偶校验实现方法是:由若干位有效信息(如一个字节),再加上一个二进制位(校验位)组成校验码,如图 2-5所示。校验位的取值(0 或 1)将使整个校验码中



图 2-5　奇偶校验码

1 的个数为奇数或偶数，所以有两种可供选择的校验规律：

奇校验——整个校验码（有效信息位和校验位）中 1 的个数为奇数。

偶校验——整个校验码中 1 的个数为偶数。

设有效信息为 $D_7D_6D_5D_4D_3D_2D_1D_0$，在发送端形成校验位 $D_{校}$。

偶校验：$D_{校}=D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$

奇校验：$D_{校}=\overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$

将校验位和有效信息位一起传送到接收端，在接收端进行校验检测。

偶校验：$P=D_{校} \oplus D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$

奇校验：$P=\overline{D_{校} \oplus D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$

若 $P=0$，则无错；若 $P=1$，则有错。

注意：奇偶校验位是按奇或偶检验规律产生的位，只有一个二进制位；而奇偶校验码共 $n+1$ 位，不仅包括奇偶校验位，还包括所有 n 位有效信息位。

除简单的奇偶校验外，还有交叉的奇偶检验，即横向、纵向同时进行奇偶校验。若一个系统传输若干个长度为 m 位的信息。如果把这些信息都编成每组 n 个信息的分组，则在这些不同的信息间，也如对单个信息一样，能够作奇偶校验。表 2-5 中列出了 n 个 m 位信息的矩阵，并以横向奇偶（HP）及纵向奇偶（VP）的形式定出奇偶校验位。

表 2-5　交叉的奇偶检验

	m 位数字				横向奇偶位
n 个字码	a_1	a_2	…	a_m	HP_1
	b_1	b_2	…	b_m	HP_2
	⋮	⋮	⋮	⋮	⋮
	n_1	n_2	…	n_m	HP_n
纵向奇偶位	VP_1	VP_2	…	VP_m	HP_{n+1}

这种校验的优点是：不仅能检测许多形式的错误，而且在给定的行或列中产生孤立的错误时，还可对该错误进行纠正。

14．汉明码校验

汉明码是广泛采用的一种有效的校验码，它实际上是一种多重奇偶校验，其实现原理是：在有效信息位中加入几个校验位形成汉明码，并把汉明码的每一个二进制位分配到几个奇偶校验组中。当某一位出错后，就会引起有关的几个校验位的值发生变化，这不但可以发现错误，还能指出错误的位置，为自动纠错提供了依据。

例如，$N=8$，则汉明码的总位数为 13 位，

$$P_5 \quad D_8 \quad D_7 \quad D_6 \quad D_5 \quad P_4 \quad D_4 \quad D_3 \quad D_2 \quad P_3 \quad D_1 \quad P_2 \quad P_1$$

5 个校验位中，$P_1 \sim P_4$ 分别处于 2^0、2^1、2^2、2^3 的位置，而 P_5 放在汉明码的最高位上。所以，$P_5 \sim P_1$ 对应的汉明码位号应分别为 H_{13}、H_8、H_4、H_2、H_1，余下的各位为有效信息位。

每一位汉明码和参与对其校验的有关校验位的对应关系见表 2-6。

表 2-6　汉明码与校验位的对应关系

汉明码位号	H_{13}	H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
信息/校验位	P_5	D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
$P_1(2^0)$			√		√		√		√		√		√
$P_2(2^1)$			√	√			√	√			√	√	
$P_3(2^2)$		√					√	√	√	√			
$P_4(2^3)$		√	√	√	√	√							
P_5	×	×		×	×			×	×		×		

从表 2-6 中可以看出,D_1 的汉明位号为 H_3,3=1+2,由位于 H_1 的 P_1 和 H_2 的 P_2 校验;D_2 的汉明位号为 H_5,5=1+4,由位于 H_1 的 P_1 和 H_4 的 P_3 校验……以此类推,可以得到:

$$P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$
$$P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$
$$P_3 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$$
$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

在上述 4 式中,D_4 和 D_7 出现了 3 次,而 D_1、D_2、D_3、D_5、D_6、D_8 仅出现了 2 次,使不同代码的汉明码的码距不等,为此,再补充一位 P_5 校验位,见表 2-6 中的×,使

$$P_5 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8$$

在这种安排下,每一位信息位都均匀地出现在 3 个 P_i 值的形成关系中。当任一位信息位发生变化时,必将引起 3 个 P_i 值跟着变化,即合法汉明码的码距都为 4。实际上,P_5 的出现仅是为了使码距相等,所以 8 位有效信息加 4 位校验位就可以了。P_1~P_4 这 4 个校验位将 12 位汉明码分为 4 组,第 i 位由校验位号之和等于 i 的那些校验位所校验。如第 11 位 D_7 由 P_1(位号为 1)、P_2(位号为 2)、P_4(位号为 8)所校验,因为 1+2+8=11。

在接收端将接收到的汉明码按如下关系进行偶校验,即

$$S_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$
$$S_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$
$$S_3 = P_3 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8$$
$$S_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8$$
$$S_5 = P_5 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8$$

将 S_5~S_1 作为指误字,可以检测和自动纠正一位错(即确定一位出错位的位置),并能发现两位错。

通常,汉明码可以分为两种,能纠正一位错的汉明码和能纠正一位错并能同时发现两位错的汉明码,两者的区别仅在于前者比后者要少一位检验位。校验位的位数 K 和信息位的位数 N 应满足下列关系:$2^K \geq N+K+1$(单纠错),$2^{K-1} \geq N+K+1$(单纠错/双检错)。上述汉明码能纠正一位错并能发现两位错,对于仅能纠正一位错的汉明码可以减少一位检验位,如前述中的 P_5。

2.3 典型例题详解

【例 2.1】 数值 $X=(ab)_{10}=(ba)_{16}$，其中 a、b 均为 1～9 的数值符号。问 a、b 各为多少？

解：因为 $X=a\times10+b=b\times16+a$，故有 $3a=5b$，在 1～9 之间只有 $a=5$，$b=3$ 满足条件，即 $X=(53)_{10}=(35)_{16}$。

【例 2.2】 设某机器数为 10001000，分别写出当其为原码表示、反码表示和补码表示时，所对应十进制真值。

解：若将机器数 10001000 看成原码，则其对应的十进制真值为 -8；若将机器数 10001000 看成反码，则其对应的十进制真值为 -119；若将机器数 10001000 看成补码，则其对应的十进制真值为 -120。

【例 2.3】 假设机器字长为 8 位，分别写出 -1 的补码用定点整数和定点小数表示的形式。

解：假设机器字长为 8 位，用补码表示。当定点整数(纯整数)时，-1 为绝对值最小的负数，其机器数表示为 11111111；当定点小数(纯小数)时，-1 为绝对值最大的负数，其机器数表示为 1.0000000。

【例 2.4】 使用 20 位数码寄存器(含 1 位符号位)能表示二进制定点整数的数值范围多大？若用 BCD 码表示十进制定点整数，其数值范围多大？

解：以原码、反码进行数据表示时，数值范围为 $-(2^{19}-1)\sim2^{19}-1$；以补码、移码进行数据表示时，数值范围为 $-2^{19}\sim2^{19}-1$。可见补码、移码表示的数据个数比原码、反码表示的数据个数多一个。

用 BCD 码表示十进制定点整数：无符号表示为 0～99999，带符号表示为 $-9999\sim9999$。

【例 2.5】 某浮点数字长 16 位，其中阶码部分 6 位(含一位阶符)，移码表示，以 2 为底；尾数部分 10 位(含一位数符，位于尾数的最高位)，补码表示，规格化。分别写出下列各题的二进制代码和其对应的真值。

(1) 非零最小正数；

(2) 最大正数；

(3) 绝对值最小负数；

(4) 绝对值最大负数。

解：(1) 非零最小正数位于数轴上正方向最接近零的位置，此时阶码为绝对值最大的负数(最小值)，尾数为规格化最小正数，其二进制浮点数的形式和真值分别为：

$$000000,\quad 0.100000000,\quad 2^{-1}\times2^{-2^5}=2^{-33}$$

(2) 最大正数位于数轴上正方向最右边的位置，此时阶码和尾数均为最大正数，其二进制浮点数的形式和真值分别为：

$$111111,\quad 0.111111111,\quad (1-2^{-9})\times2^{2^5-1}=(1-2^{-9})\times2^{31}$$

(3) 绝对值最小负数位于数轴上负方向最接近零的位置，此时阶码为绝对值最大的负数，尾数为规格化的绝对值最小负数，其二进制浮点数的形式和真值分别为：

000000, 1.011111111, $-(2^{-1}+2^{-9})\times 2^{-2^5}=-(2^{-1}+2^{-9})\times 2^{-32}$

(4) 绝对值最大负数位于数轴上负方向最左边的位置,此时阶码为最大正数,尾数为绝对值最大负数,其二进制浮点数的形式和真值分别为:

111111, 1.000000000, $-1\times 2^{2^5-1}=-2^{31}$

注意:由于阶码用移码表示,所以浮点数二进制代码形式的阶码部分最高位应与补码时相反(此时假设偏置值为 2^5)。

【例 2.6】 设 32 位长的浮点数,其中阶符 1 位,阶码 7 位,数符 1 位,尾数 23 位。分别写出机器数采用原码和补码表示时,所对应的最接近 0 的负数。

解:最接近 0 的负数就是绝对值最小的负数。此时,该数的阶码为负,且绝对值最大;该数的尾数为负,且绝对值最小。由于题目中并未指出数据是否为规格化的数据,这里可以假定浮点数据为规格化数据,也可以假定浮点数据为非规格化数据。

采用原码表示时,最接近于 0 的负数的阶码为 $-(2^7-1)=-127$,尾数为 -2^{-23}(非规格化数)或 -2^{-1}(规格化数)。所以,该数为 $-2^{-23}\times 2^{-127}$(非规格化数)或 $-2^{-1}\times 2^{-127}$(规格化数)。

采用补码表示时,最接近于 0 的负数的阶码为 $-2^7=-128$,尾数为 -2^{-23}(非规格化数)或 $-(2^{-1}+2^{-23})$(规格化数)。所以,该数为 $-2^{-23}\times 2^{-128}$(非规格化数)或 $-(2^{-1}+2^{-23})\times 2^{-128}$(规格化数)。

【例 2.7】 设浮点数字长 16 位,其中阶码 5 位(含 1 位阶符),以 2 为底,补码表示;尾数 11 位(含 1 位数符),补码表示,判断下列各十进制数能否表示成规格化浮点数。若可以,请表示。

(1) 3.5;

(2) 79/512;

(3) -10^{-4};

(4) 10^{10}。

解:(1) $3.5=(11.1)_2=2^2\times(0.111)_2$,其规格化浮点数表示为:

0 0010 0 1110000000

(2) $79/512=2^{-9}\times(1001111)_2=2^{-2}\times(0.1001111)_2$,其规格化浮点数表示为:

1 1110 0 1001111000

(3) $-10^{-4}=(-0.0001)_{10}$,若转换为二进制,前面应当有 13 个头 0,约等于 $-2^{-13}\times(0.1101000110)_2$,其规格化浮点数表示为:

1 0011 1 0010111010

(4) $10^{10}=10^{(9+1)}=10^9\times 10>10^9\approx 2^{30}$,而这种格式的浮点数能表示的最大正数仅为 $(1-2^{-10})\times 2^{15}$,所以,此数已超过浮点数的表示范围,不能表示成规格化浮点数。

【例 2.8】 浮点数的阶码为什么通常采用移码?

解:浮点数的阶码通常采用移码的主要原因有两个:

(1) 便于比较浮点数的大小。移码的大小直观反映了真值的大小,不必考虑符号问题,阶码大的,其对应的真值就大,阶码小的,对应的真值就小。

(2) 简化机器中的判零电路。当阶码全为 0,尾数也全为 0 时,表示机器零。当浮点数结果的阶码 $<-2^n$,而尾数 $\neq 0$ 时,将这个数据当作机器零处理。如果使用移码表示阶码,

则阶码的形式为00…00；如果使用补码表示阶码，则阶码的形式为10…00。

【例2.9】 设计一个浮点数据格式，用尽量少的位数满足以下要求：

(1) 数值范围为$-1.0\times10^{38}\sim-1.0\times10^{-38}$和$1.0\times10^{-38}\sim1.0\times10^{38}$；

(2) 精度为表示7位十进制数据(相对精度)；

(3) 用全0表示数据0。

解：(1) 因为$2^{10}>10^3$，可得$2^{120}=(2^{10})^{12}>(10^3)^{12}=10^{36}$，又因为$2^7>10^2$，由此可知$2^{127}>10^{38}$；同理$2^{-127}<10^{-38}$，所以阶码取8位，其数值范围为$-128\sim127$。

(2) 因为$2^{23}\approx10^7$，所以可取尾数23位，加上符号位和阶码，共32位。

(3) 用全0表示数据0，则阶码用移码表示，尾数用补码表示。

【例2.10】 写出下列十进制数的IEEE 754短浮点数编码。

(1) 0.15625；

(2) -5。

解：(1) 0.15625转换成二进制数值为0.00101，在IEEE 754中，其规格化表示为1.01×2^{-3}，$E=127-3=124$。

IEEE 754短浮点数编码为：

0　01111100　01000000000000000000000

(2) -5转换成二进制数值为-101，在IEEE 754中，其规格化表示为1.01×2^2，$E=127+2=129$。

IEEE 754短浮点数编码为：

1　10000001　01000000000000000000000

注意：*尾数的最高位1是隐含的。*

【例2.11】 若短浮点数IEEE 754编码为1011 1111 0100 0000 0000 0000 0000 0000，则其代表的十进制数为多少？

解：短浮点数IEEE 754编码的格式为：数符1位，阶码8位(移码表示)，尾数23位。将本题编码按格式展开后为：

1　01111110　10000000000000000000000

阶码真值 $=E-127=01111110-01111111=-1$

尾数(包括隐含位) $=1.10000000000000000000000=1.1$

所以，其代表的十进制数为：

$$-(1.1)\times2^{-1}=-(0.11)_2=-(0.75)_{10}$$

【例2.12】 以2为基数，有1位符号位、4位阶码和8位二进制尾数代码的浮点数，阶码采用移码表示，求数值表示范围及可表示的数据个数。

解：假设尾数与符号位共同构成原码。

最大规格化尾数：$1-2^{-8}$。

最小规格化尾数为2^{-1}。因为规格化尾数要求尾数的最高数位必须是1。

最大阶码：$2^3-1=7$。

最小阶码：$-2^3=-8$。

将最大规格化尾数乘以2的最大阶码次方，就得到最大正值：$(1-2^{-8})\times2^7$。

将最小规格化尾数乘以2的最小阶码次方，就得到最小正值：$2^{-1}\times2^{-8}=2^{-9}$。

同理可得最大负值(绝对值最小的负数)：$-2^{-1}\times2^{-8}=-2^{-9}$。

最小负值(绝对值最大的负数)：$-(1-2^{-8})\times2^{7}$。

规格化尾数个数：2^8，其中 2^7 个正值，2^7 个负值。

可表示的数据个数：$2^{12}+1$ 个。阶码有 2^4 种组合，共有 $2^4\times2^7=2^{11}$ 个正值，同样也有 2^{11} 个负值，再加上机器零。

【例 2.13】 计算机存储程序概念的特点之一，是把数据和指令都作为二进制信号看待。今有一计算机字长 32 位($D_{31}\sim D_0$)，数符位是第 31 位。

对于二进位 1000 1111 1110 1111 1100 0000 0000 0000，

(1) 表示一个补码整数，其十进制值是多少？

(2) 表示一个无符号整数，其十进制值是多少？

(3) 表示一个 IEEE 754 标准的单精度浮点数，其值是多少？

解：(1) 表示一个补码整数，

最高位为符号位，其他 31 位为数值位。其对应的真值二进制表示为：

$$-111\ 0000\ 0001\ 0000\ 0100\ 0000\ 0000\ 0000$$

其十进制值是：

$$-(2^{30}+2^{29}+2^{28}+2^{20}+2^{14})$$

(2) 表示一个无符号整数，

全部 32 位均为数值位，其十进制值是：

$$2^{31}+2^{27}+2^{26}+2^{25}+2^{24}+2^{23}+2^{22}+2^{21}+2^{19}+2^{18}+2^{17}+2^{16}+2^{15}+2^{14}$$

(3) 表示一个 IEEE 754 标准的单精度浮点数：

$$\underline{1};\ \underline{00011111};\ \underline{11011111100000000000000}$$

↑ 数符　↑ 阶码　↑ 尾数

因为阶码为 00011111，对应十进制数为 31。

所以 IEEE 754 标准中的阶码用移码表示，单精度浮点数的偏置值 127，所以阶码的十进制真值为：

$$31-127=-96$$

因为尾数为 1.11011111100000000000000。

所以 IEEE 754 标准中的尾数用原码表示，且采用隐含尾数最高数位为 1 的方法，隐含的 1 是一位整数(即位权为 2^0)。所以尾数真值为：

$$2^0+2^{-1}+2^{-2}+2^{-4}+2^{-5}+2^{-6}+2^{-7}+2^{-8}+2^{-9}$$

因为数符＝1，表示这个浮点数是个负数。

所以单精度浮点数的真值为：

$$-(2^0+2^{-1}+2^{-2}+2^{-4}+2^{-5}+2^{-6}+2^{-7}+2^{-8}+2^{-9})\times2^{-96}$$

【例 2.14】 汉字的区位码、国标码和机内码有什么区别？已知汉字“春”的国标码为 343AH，试分别写出它的区位码和机内码。

解：汉字区位码实际上是一种输入码，区位码定长 4 位十进制数，前 2 位表示区号，后 2 位表示位号。

汉字国标码(GB 码)又称为汉字交换码，每个汉字都用两个字节表示，通常写成 4 位十

六进制数形式。

汉字区位码和国标码有一一对应关系，国标码中所包含的全部汉字被分为94个区，每个区中包含94个汉字(位)，区和位组成一个二维数组，每个汉字在数组中对应一个唯一的区位码。

汉字机内码是汉字在计算机内部的编码，机内码也是两字节长的代码。

汉字机内码是在国标码的基础上，每个字节的最高位上加1后得到的。机内码的出现是为了避免当系统中同时存在ASCII码和汉字国标码时产生二义性。

因为汉字"春"的国标码为343AH，根据

汉字国标码 = 汉字区位码(十六进制) + 2020H

汉字机内码 = 汉字国标码 + 8080H

可以得出：

"春"的区位码(十六进制)=343AH－2020H=141AH，然后将141AH转换成十进制数，故"春"的区位码写作20－26。

"春"的机内码=343AH＋8080H=B4BAH。

【例2.15】 判断如表2-7所示一个BCD码的编码系统是有权码还是无权码，写出判断的推导过程。

表2-7　BCD码

十进制数	BCD编码
0	0000
1	0111
2	0110
3	0101
4	0100
5	1011
6	1010
7	1001
8	1000
9	1111

解：设该BCD码从左至右各位分别为A、B、C、D，且假定其为有权码，则

从数值8的编码1000，可求得A的位权为8；

从数值4的编码0100，可求得B的位权为4；

从数值6的编码1010，可求得C的位权为－2；

从数值7的编码1001，可求得D的位权为－1。

最后用A、B、C、D各位的位权分别来验证其他数值的编码值，结果都正确，说明这种BCD码(8 4 －2 －1码)是一种有权码。

【例2.16】 某一数据为10101010，若采用奇校验，其校验位是什么？

解：若采用奇校验，则有

$$\text{奇校验位} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$$

$$= \overline{1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0} = 1$$

现在8位数据10101010中1的个数有4个(偶数),所以奇校验位为1。

【例2.17】 由6个字符的7位ASCII编码排列,再加上横向、纵向奇偶校验位构成如表2-8所示的矩阵(最后一列为横向奇偶校验位,最后一行为纵向奇偶校验位)。请分别写出X_1~X_{12}代表的数字(0或1),以及Y_1、Y_2代表的字符。

表2-8　6个字符的交叉检验矩阵

字　符	7位ASCII码							HP
3	0	X_1	X_2	0	0	1	1	0
Y_1	1	0	0	1	0	0	X_3	1
+	X_4	1	0	1	0	1	1	0
Y_2	0	1	X_5	X_6	1	1	1	1
D	1	0	0	X_7	1	0	X_8	0
=	0	X_9	1	1	1	X_{10}	1	1
VP	0	0	1	1	1	X_{11}	1	X_{12}

解: 从ASCII码左起第5列可知纵向为偶校验,据此可求出$X_4=0$,$X_{12}=1$。

根据$X_4=0$可知横向也是偶校验,可求出$X_3=1$,$X_{11}=1$。

根据$X_{11}=1$,可求出$X_{10}=0$;根据$X_{10}=0$,可求出$X_9=1$;根据$X_9=1$,可求出$X_1=1$;根据$X_1=1$,可求出$X_2=1$;根据$X_2=1$,可求出$X_5=1$;根据$X_5=1$,可求出$X_6=0$;根据$X_6=1$,可求出$X_7=0$;根据$X_7=0$,可求出$X_8=0$。

故X_1~X_{12}的数字依次为111010001011。

由字符Y_1的ASCII码1001001=49H知道,Y_1即是字母"I"(由D的ASCII码是1000100=44H推得);由字符Y_2的ASCII码0110111=37H知道,Y_2即是数字"7"(由"3"的ASCII码是0110011=33H推得)。

【例2.18】 请写出数据10110100110的汉明码,用4位检验位,采用偶检验。

解: 本题中只有4位检验位,检验位应位于汉明码的1、2、4、8的位置上,这种汉明码具有单纠错功能。根据汉明码的定义有:

$D_{11}\quad D_{10}\quad D_9\quad D_8\quad D_7\quad D_6\quad D_5\quad \underline{\boldsymbol{P_4}}\quad D_4\quad D_3\quad D_2\quad \underline{\boldsymbol{P_3}}\quad D_1\quad \underline{\boldsymbol{P_2}}\quad \underline{\boldsymbol{P_1}}$

$P_1=D_1\oplus D_2\oplus D_4\oplus D_5\oplus D_7\oplus D_9\oplus D_{11}=0\oplus 1\oplus 0\oplus 0\oplus 0\oplus 1\oplus 1=1$

$P_2=D_1\oplus D_3\oplus D_4\oplus D_6\oplus D_7\oplus D_{10}\oplus D_{11}=0\oplus 1\oplus 0\oplus 1\oplus 0\oplus 0\oplus 1=1$

$P_3=D_2\oplus D_3\oplus D_4\oplus D_8\oplus D_9\oplus D_{10}\oplus D_{11}=1\oplus 1\oplus 0\oplus 1\oplus 1\oplus 0\oplus 1=1$

$P_4=D_5\oplus D_6\oplus D_7\oplus D_8\oplus D_9\oplus D_{10}\oplus D_{11}=0\oplus 1\oplus 0\oplus 1\oplus 1\oplus 0\oplus 1=0$

所以,数据10110100110的汉明码为1 0 1 1 0 1 0 $\underline{0}$ 0 1 1 $\underline{1}$ 0 $\underline{1}$ $\underline{1}$。加下划线处为检验位。

*【例2.19】 float型数据通常用IEEE 754单精度浮点数格式表示。若编译器将float型变量x分配在一个32位浮点寄存器FR1中,且$x=-8.25$,则FR1的内容是________。

A. C104 0000H　　B. C242 0000H　　C. C184 0000H　　D. C1C2 0000H

解: A。

分析: 首先将十进制数转换为二进制数-1000.01,接着把它写成规格化形式-1.00001×2^3(按IEEE 754标准),然后计算阶码的移码(偏置值+阶码真值)为130,最后

短浮点数代码：符号位=1，阶码=10000010，尾数 00001000000000000000000，写成十六进制为 C1040000H。

*【例 2.20】 假定编译器规定 int 和 short 类型长度分别为 32 位和 16 位，执行下列 C 语言语句：

```
unsigned short  x= 65530;
unsigned int    y= x;
```

得到 y 的机器数为________。

A. 0000 7FFAH　　B. 0000 FFFAH　　C. FFFF 7FFAH　　D. FFFF FFFAH

解：B。

分析：x 和 y 均为无符号数，其中 x 为 16 位，y 为 32 位，将 16 位无符号数转化成 32 位无符号数，前面要补零。因为 x=65530=FFFAH，所以 y=0000 FFFAH。

*【例 2.21】 float 类型（即 IEEE 754 单精度浮点数格式）能表示的最大正整数是________。

A. $2^{126}-2^{103}$　　B. $2^{127}-2^{104}$　　C. $2^{127}-2^{103}$　　D. $2^{128}-2^{104}$

解：D。

分析：IEEE 754 单精度浮点数能表示的最大正整数（十六进制）为 7F7FFFFFH，其数值为 $1+(1-2^{-23})\times 2^{254-127}=(2-2^{-23})\times 2^{127}=2^{128}-2^{104}$。

*【例 2.22】 某数采用 IEEE 754 单精度浮点数格式表示为 C640 0000H，则该数的值是________。

A. -1.5×2^{13}　　B. -1.5×2^{12}　　C. -0.5×2^{13}　　D. -0.5×2^{12}

解：A。

分析：将单精度浮点数 C640 0000H 转换成二进制 1100 0110 0100 0000 0000 0000 0000 0000，分离出符号位、阶码和尾数三部分。符号位=1，表示结果是一个负数；阶码=10001100，阶码真值=140−127=13；尾数=10000000000000000000，由于尾数隐含了一个1，所以尾数为 1.5（十进制）。最终的结果是 -1.5×2^{13}。

此题很容易误选 C，这是因为忽略了尾数隐含 1 的问题。

*【例 2.23】 用汉明码对长度为 8 位的数据进行检/纠错时，若能纠正一位错，则检验位数至少为________。

A. 2　　B. 3　　C. 4　　D. 5

解：C。

分析：汉明码不是只有一种，若能检测和自动校正一位错，此时校验位的位数 K 和信息位的位数 N 应满足下列关系 $2^K\geqslant N+K+1$。现汉明码的信息位 N 为 8 位，则校验位 K 的位数至少为 4 位。

*【例 2.24】 float 型数据通常用 IEEE 754 单精度浮点数表示。假定两个 float 型变量 x 和 y 分别存放在 32 位寄存器 f1 和 f2 中，若(f1)=CC90 0000H，(f2)=B0C0 0000H，则 x 和 y 之间的关系为________。

解：A。

分析：根据 f1 和 f2 两个寄存器的内容，可以认定 x 和 y 两数均为负数（数符位为 1），

所以选项B、D可以排除;接下来看两数的阶码(移码表示),可见 x 的阶码大于 y 的阶码,因为是负数,所以 $x<y$。

A. $x<y$ 且符号相同　　B. $x<y$ 且符号不同

C. $x>y$ 且符号相同　　D. $x>y$ 且符号不同

2.4　同步测试习题及解答

2.4.1　同步测试习题

一、填空题

1. 设 $X=-69$,$n=8$(含符号位),则 X 的原码为________,X 的补码为________,X 的移码为________。

2. 十进制数64.5所对应的二进制数表示为________,8421码表示为________。

3. 已知 $X=-11$,则 X 的二进制数表示形式是________,十六进制表示形式是________,8421码为________,原码为________,补码为________。

4. 设机器字长为8位,$X=78$,$Y=-97$,则

$[X]_{原}=$________B,$[X]_{补}=$________B。

$[X]_{原}=$________B,$[X]_{补}=$________B。

5. 阶码8位(最左一位为符号位),用移码表示,尾数为24位(最左一位为符号位),用规格化补码表示,则它能表示的最大正数的阶码为________,尾数为________;绝对值最小的负数的阶码为________,尾数为________。(以上回答用二进制书写)

6. $[-0]_{反}$ 表示为________。

7. 8位补码定点整数所能表示的绝对值最大的负数(即最负的数)的值为________。

8. 补码定点小数所能表示的绝对值最大负数的值为________。

9. 当浮点数的尾数为补码时,其为规格化数应满足的条件为________。

10. $(1978)_{10}=($________$)_{2421码}$。

11. 已知某个汉字的国标码为3547H,其机内码为________H。

二、选择题

1. 零的原码可以用以下哪个代码表示________。

A. 11111111　　B. 10000000　　C. 01111111　　D. 11000000

2. 9位原码能表示的数据个数是________。

A. 10　　B. 9　　C. 511　　D. 512

3. n 位二进制定点整数表示的最大值是________。

A. 2^n　　B. 2^n-1　　C. 2^{n-1}　　D. $2^{n-1}-1$

4. 定点8位字长的字,采用2的补码表示时,一个字所表示的整数范围是________。

A. $-128\sim127$　　B. $-129\sim128$　　C. $-127\sim127$　　D. $-128\sim128$

5. 一个8位二进制整数,若采用补码表示,且由4个1和4个0组成,则最小值为________。

A. -120　　B. -7　　C. -112　　D. -121

6. 已知$[X]_{补}=1.X_1X_2X_3X_4X_5$，若要 $X>-1/2$，$X_1X_2X_3X_4X_5$ 应满足________。

A. X_1 必须为 1，$X_2X_3X_4X_5$ 至少有一个 1

B. X_1 必须为 1，$X_2X_3X_4X_5$ 任意

C. X_1 必须为 0，$X_2X_3X_4X_5$ 至少有一个 1

D. X_1 必须为 0，$X_2X_3X_4X_5$ 任意

7. 在定点机中，下列说法错误的是________。

A. 除补码外，原码和反码不能表示－1

B. ＋0 的原码不等于－0 的原码

C. ＋0 的反码不等于－0 的反码

D. 对于相同的机器字长，补码比原码和反码能多表示一个负数

8. 设寄存器内容为 11111111，若它等于＋127，则为________。

A. 原码　B. 补码　C. 反码　D. 移码

9. 在规格化浮点数表示中，保持其他方面不变，将阶码部分的移码表示改为补码表示，将会使数的表示范围________。

A. 增大　B. 减少　C. 不变　D. 以上都不对

10. 若 9BH 表示移码，其对应的十进制数是________。

A. 27　B. －27　C. －101　D. 101

11. 在浮点数中，当数据的绝对值太小，以至于小于所能表示的数据时，称为浮点数的________。

A. 下溢　B. 负下溢　C. 负溢　D. 正下溢

12. 设浮点数阶码的基数是 8，下列浮点数尾数中规格化数是________。

A. 11.111000　B. 00.000111　C. 11.101010　D. 11.111101

13. 目前在小型和微型计算机里普遍采用的字符编码是________。

A. BCD 码　B. 十六进制代码　C. ASCII 码　D. 汉明码

14. 已知大写英文字母 A 的 ASCII 码为 41H，现字母 F 被存放在某个存储单元中，若采用偶校验（假设最高位作为校验位），则该存储单元中存放的十六进制数据是________。

A. 46H　B. C6H　C. 47H　D. C7H

15. 汉字“啊”的十进制区位码为“16－01”，它的十六进制机内码为________。

A. 1601H　B. 9081H　C. B0A1H　D. B081H

16. 某数在计算机中用 8421 码表示为 0111 1000 1001，其真值为________。

A. 789　B. 789H　C. 1929　D. 11110001001B

17. 在计算机中，用 BCD 码表示 0～99 之间的数需要用________，一个字节存放________个一位的 BCD 码。

A. 5，1　B. 6，3　C. 7，2　D. 8，2

18. 采用十进制字符串数据表示时，－123 的前分隔数字串、后嵌入数字串和压缩的十进制数串的字节长度分别是________。

A. 4，4，2　B. 4，3，2　C. 4，4，3　D. 4，3，3

19. 假定下列字符码中有奇偶检验位，但没有数据错误，采用奇检验的字符码是________。

A. 11001010　　B. 11010111　　C. 11001100　　D. 11001011

20. 若信息码字为11100011，生成多项式 $G(x)=x^5+x^4+x+1$，则计算出的CRC校验码为________。

A. 1110001101101　　B. 1110001111010

C. 11100011001101　　D. 111000110011010

三、判断题

1. 若$[X]_补>[X]_补$，则$|X|>|Y|$。　（　　）

2. 浮点数通常采用规格化数来表示，规格化即指其尾数的第1位应为1的浮点数。
（　　）

3. 浮点数的取值范围由阶码的位数决定，而浮点数的精度由尾数的位数决定。（　　）

4. 8421码就是二进制数。　（　　）

四、综合题

1. 在整数定点机中，若寄存器的内容为80H，当它分别代表原码、补码、反码、移码和无符号数时，所对应的十进制数值各为多少？

2. 分别用16位带符号二进制数(补码)及4位十六进制数表示十进制数+146D及−31549D。

3. 按下述规定格式(阶符1位，阶码7位，尾符1位，尾数23位)，写出真值为$-\frac{23}{4096}$的补码规格化浮点数形式。

4. 某机字长32位，在浮点表示时，阶码占8位，尾数占24位，各包含一位符号位，问：

(1) 带符号定点小数的最大表示范围是多少？

(2) 带符号定点整数的最大表示范围是多少？

(3) 浮点表示时，最大的正数是多少？

(4) 浮点表示时，最大的负数是多少？

(5) 浮点表示时，最小的规格化正数是多少？

5. 使用32位浮点二进制数，8位(含1位符号位)为用补码表示的阶码，24位(含1位符号位)为补码表示的规格化尾数，试指出它所表示的最大正数与最小正数的数据格式。

6. 设浮点数的格式如下：

第15位：符号位；

第14～8位：阶码，采用补码表示；

第7～0位：尾数，与符号位一起采用规格化的补码表示，基数为2。

问：(1) 它能表示的数值范围是什么？

(2) 它能表示的最接近于0的正数和负数分别是什么？

(3) 它共能表示多少个数值？

7. 某浮点数字长16位，问：

(1) 写出十进制−11.4的规格化浮点数形式表示，具体要求是阶码为4位二进制移码(其偏置值为2^3)，尾数为12位原码(含数的符号)；

(2) 写出上述格式定义的规格化浮点数所能表示的最大与最小的正数和绝对值最大与最小的负数的值；

(3) 说明上述格式定义的浮点数的机器零。

8. 下列 IEEE 单精度浮点数所表示的十进制数分别是多少？

(1) 1011 1101 0100 0000 0000 0000 0000 0000；

(2) 0101 0101 0110 0000 0000 0000 0000 0000；

(3) 1100 0001 1111 0000 0000 0000 0000 0000；

(4) 0011 1010 1000 0000 0000 0000 0000 0000。

9. 对于下列每个 IEEE 单精度数值，请解释它们所表示的是哪一种数字类型（规格化数、非规格化数、无穷大、0 或 NaN）。当它们表示某个具体数值时，请给出该数值。

(1) 0111 1111 1000 1111 0000 1111 0000 0000；

(2) 0000 0000 0000 0000 0000 0000 0000 0000；

(3) 0100 0010 0100 0000 0000 0000 0000 0000；

(4) 1000 0000 0100 0000 0000 0000 0000 0000；

(5) 1111 1111 1000 0000 0000 0000 0000 0000。

2.4.2 同步测试习题解答

一、填空题

1. 11000101，10111011，00111011。

2. 1000000.1，0110 0100.0101。

3. －1011B，－BH，0001 0001 1101，10001011，11110101。

4. 01001110，01001110，11100001，10011111。

5. 11111111，01111111111111111111111，00000000，10111111111111111111111。

注意：因为阶码部分用移码表示，所以最大值用连续的 8 个 1 表示，最小值用连续的 8 个 0 表示。

6. 11111111（假设字长为 8 位）。

7. －128（十进制）。

8. －1（十进制）。

9. $m_s \oplus m_1 = 1$。

10. 0001 1111 1101 1110。

11. B5C7。

二、选择题

1. B。在原码表示中，$[+0]_原=00000000$，$[-0]_原=10000000$。

2. C。能表示的数据个数$=2^9-1=511$。

3. D。n 位二进制定点整数，数值位只有 $n-1$ 位，所以最大值为 $2^{n-1}-1$。

4. A。8 位补码的表示范围为$-2^7\sim+(2^7-1)$，其十进制真值为$-128\sim+127$。

5. D。补码负数的特点是数值位对应的真值越小，其绝对值越大，即负得越多。所以，由 4 个 1 和 4 个 0 组成的补码数中，最小的补码表示为 10000111，即真值为－121。

6. A。当 $X=-1/2$ 时，$[X]_补=1.10000$。在 $-1/2<X<0$ 范围内，$X_1=1$，$X_2+X_3+X_4+X_5=1$。

7. A。对于定点小数来说，A 的说法是正确的，但对于定点整数来说，A 的说法就不正确了。因为假设机器字长为 8 位，在整数表示时，$[-1]_原=1,0000001$，$[-1]_补=1,1111111$，$[-1]_反=1,1111110$。

8. D。对于偏置值为 2^n 的移码，同一数值的移码和补码除最高位相反外，其他各位相同。因为$[+127]_补=01111111$，所以$[+127]_移=11111111$。

9. C。因为将阶码部分的移码表示改为补码表示，并不会使数的表示范围发生变化，只会使阶码的表示形式发生变化。

10. A。移码表示 9BH=10011011，则补码表示为 00011011，对应十进制真值为 27。

11. A。当数据的绝对值太小，以至于小于阶码所能表示的数(阶码下溢)时，则称为浮点数的下溢，包括正下溢和负下溢。

12. C。当阶码是以 8 为底时，只要尾数满足 $\frac{1}{8}\leqslant M<1$ 或 $-1\leqslant M<\frac{1}{8}$ 就是规格化数(补码时)。所以判断规格化时，只要尾数的数值部分的最高 3 位中有一位与符号位不同即可。

13. C。字符编码方式有很多种，现目前用的最广泛的是 ASCII 码。

14. B。英文字母 F 的 ASCII 码应为 46H=1000110B。标准的 ASCII 码为 7 位，在 7 位数前面增加 1 位校验位。按照偶校验规则，偶校验位为 1。存储单元中存放的是整个校验码(包括校验位和信息位)应为 11000110B=C6H。

15. C。区位码 16－01(十进制)=1001H，国标码=1001H+2020H=3021H，机内码=3021H+8080H=B0A1H。

16. A。8421 码用 4 位二进制编码表示 1 位十进制数。

17. D。2 位十进制数的 BCD 码需要 8 位二进制表示，一个字节中可以放两个 BCD 码。

18. B。存放这个十进制数串，前分隔方式需要 4 个字节，后嵌入方式需要 3 个字节，压缩的十进制数方式需要 2 个字节。

19. D。正确的奇检验码中 1 的个数是奇数个。

20. B。首先将信息码字左移 5 位，除以生成多项式 $G(x)$，即 1110001100000÷110011，得到余数为 11010，然后将余数拼接在信息码字的后面。

三、判断题

1. ×。仅当 X、Y 均为正数时，满足$[X]_补>[X]_补$，则$|X|>|Y|$，其他情况不满足。

2. ×。原码规格化后，正数为 0.1××…×的形式，负数为 1.1××…×的形式。补码规格化后，正数为 0.1××…×的形式，负数为 1.0××…×的形式。

3. √。

4. ×。8421 码是十进制数的编码。

四、综合题

1. －0，－128，－127，0，128。

2. 0000000010010010，0092H，1000010011000011，84C3H。

3. 首先将十进制数 $-\frac{23}{4096}$ 转换成二进制数，转换使用一些技巧可大大节省时间。

$-\frac{23}{4096}=-23\times2^{-12}$ 转换成二进制数 -10111×2^{-12}。

并写成规格化形式 -0.10111×2^{-7}。

若阶码和尾数均用补码表示，则此浮点数的形式为：

11111001;1.0100100000000000000000。

4.（1）$-1\sim1-2^{-31}$。

（2）$-2^{31}\sim2^{31}-1$。

（3）$(1-2^{-23})\times2^{127}$。

（4）$-(2^{-1}+2^{-23})\times2^{-128}$，此处所说的最大的负数就是绝对值最小的负数。

（5）$2^{-1}\times2^{-128}$。

5. 采用的数据格式为：阶符；阶码；数符.尾数。

最大正数的数据格式为：0;1111111;0.11111111111111111111111。

最小正数的数据格式为：1;0000000;0.10000000000000000000000（规格化）。

6.（1）$-1\times2^{63}\sim(1-2^{-8})\times2^{63}$。

（2）$2^{-1}\times2^{-64}$，$-(2^{-1}+2^{-8})\times2^{-64}$。

（3）因为浮点数的尾数必须是规格化的，所以它共能表示 $2^{15}=32768$ 个数值。

7. 根据题意可知，阶码部分为7位，尾数部分为9位，各包含1位符号位。

（1）首先将十进制数转换成二进制数：$-11.4=-1011.0110011\cdots$，并用规格化形式表示：$-0.10110110011\times2^4$。

阶码用4位二进制移码，尾数用12位原码（含数的符号），则浮点数的形式：1100；1.10110110011。

（2）最大正数：$(1-2^{-11})\times2^7$。

最小正数：$2^{-1}\times2^{-8}$。

绝对值最大负数：$-(1-2^{-11})\times2^7$。

绝对值最小负数：$-2^{-1}\times2^{-8}$。

（3）由于阶码采用移码表示，此浮点数的机器零用16位全0表示。

8.（1）符号位＝1，阶码字段＝01111010B＝122D，阶码真值＝122－127＝－5，尾数字段＝100 0000 0000 0000 0000 0000B。十进制数值为：

$$-(1.1)_2\times2^{-5}=-0.046875$$

（2）符号位＝0，阶码字段＝10101010B＝170D，阶码真值＝170－127＝43，尾数字段＝110 0000 0000 0000 0000 0000B。十进制数值为：

$$(1.11)_2\times2^{43}=1.539\times10^{13}\quad(\text{表示为 4 位有效数字形式})$$

（3）同理，十进制数值为：

$$-(1.111)_2\times2^4=-30$$

（4）同理，十进制数值为：

$$(1.0)_2\times2^{-10}=0.0009766\quad(\text{表示为 4 位有效数字形式})$$

9. 根据表2-3可见，

（1）由于阶码字段全部为1，并且尾数字段为非0，所以它表示 N_aN。

（2）由于符号位为0，阶码字段和尾数字段均为全0，所以它表示＋0。

（3）由于阶码字段既不为全0，也不为全1，所以它表示一个规格化数，其实际值为 $(1.1)_2\times2^5=48$。

（4）由于阶码字段为全0，尾数字段不全为0，所以它表示一个非规格化数，其实际值为 $-(0.1)_2\times2^{-126}=-2^{-127}=-5.877\times10^{-39}$（表示为4位有效数字形式）。

（5）由于符号位为1，阶码字段为全1，尾数字段为全0，所以它表示负无穷大。

第 3 章 指令系统

3.1 基本内容摘要

- 指令格式
  - 机器指令的基本格式
  - 地址码结构
    三、二、一、零地址指令的特点。
  - 指令的操作码
    规整型编码；
    非规整型编码(扩展操作码)。
- 寻址技术
  - 编址方式
    编址与编址单位；
    指令中地址码的位数。
  - 指令寻址和数据寻址
  - 基本的数据寻址方式
    立即寻址；
    寄存器寻址；
    直接寻址；
    间接寻址；
    寄存器间接寻址；
    变址寻址；
    基址寻址；
    相对寻址；
    页面寻址。
  - 变型或组合寻址方式
- 堆栈与堆栈操作
  - 堆栈结构
    寄存器堆栈；
    存储器堆栈。

◆ 堆栈操作

● 指令类型

◆ 数据传送类指令

◆ 运算类指令

◆ 程序控制类指令

转移指令；

子程序调用指令；

返回指令。

◆ 输入输出类指令

独立编址的 I/O；

统一编址的 I/O。

● 指令系统的发展

◆ 从复杂指令系统到精简指令系统

◆ VLIW 和 EPIC

3.2 重点难点梳理

1. 机器指令的长度

一条机器指令通常可以分成操作码和地址码两部分。指令的长度取决于操作码的长度、地址码的个数以及每个地址的长度，任何一条指令构成一个“指令字”。指令字长与机器字长是两个不同的概念，两者之间没有固定的关系。早期的计算机多采用定长指令字结构，即所有指令的长度都是相等的，并且指令字长和机器字长相同。随着计算机技术的发展，现代计算机多采用变长指令字结构，即指令字长既可以与机器字长相等，也可以大于或小于机器字长。若指令字长等于机器字长的指令称为单字长指令，指令字长等于半个机器字长的指令称为半字长指令，指令字长等于两个机器字长的指令称为双字长指令……例如，Intel 8086 的机器字长为 16 位，指令的长度为 1～6 个字节，最短的指令只有 8 位(半字长指令)，最长的指令为 48 位(3 字长指令)。

2. 不同地址数指令的区别

对于双操作数运算类指令来说，每条指令中都需要包含以下 4 个地址信息：

(1) 第一操作数地址 A_1；

(2) 第二操作数地址 A_2；

(3) 操作结果存放地址 A_3；

(4) 下条将要执行指令的地址 A_4。

这些地址信息可以明显的给出，称为显地址；也可以依照某种事先的约定，用隐含的方式给出，称为隐地址。

大多数计算机中用程序计数器(PC)指出下一条将要执行指令的地址，其余 3 个地址的处理方式有下面 4 种(以双操作数的加法指令为例)。

1) 三地址指令

三地址指令表示指令中有 3 个显地址，指令的含义为：

$$(A_1)+(A_2)\rightarrow A_3$$

执行一条三地址的加法指令需要访问4次主存。第一次取指令本身,第二次取被加数,第三次取加数,第四次保存结果。

2) 二地址指令

二地址指令表示指令中有两个显地址,让第一操作数地址同时兼作结果存放地址(目的地址),指令的含义为:

$$(A_1)+(A_2)\rightarrow A_1$$

执行一条二地址的加法指令同样需要访问4次主存。

3) 一地址指令

一地址指令只有一个显地址,参加运算的另一个操作数来自累加寄存器Acc。指令的含义为:

$$(Acc)+(A_1)\rightarrow Acc$$

执行一条一地址的加法指令只需要访问两次主存。第一次取指令本身,第二次取操作数。由于被操作数和结果都放在累加寄存器中,所以读取和存入都不需要访问主存。

4) 零地址指令

零地址指令中只有操作码字段,没有显地址。零地址的加法指令仅用在堆栈计算机中,堆栈计算机没有一般计算机中必备的通用寄存器,操作数和结果都保存在堆栈中。通常,参加加法运算的两个操作数隐含地从堆栈顶部弹出,送到运算器中进行运算,运算的结果再隐含地压入堆栈。

如果采用存储器堆栈,执行一条零地址的加法指令仍需要访问4次主存,这是因为存储器堆栈就是主存的一部分;而如果采用寄存器堆栈,执行一条零地址的加法指令只需要在取指令的时候访问一次主存即可。

3. 二地址指令的分类

前面所提到的地址都是指存储器的地址,如二地址M-M型指令,两个操作数均在主存中,执行一条运算类指令需要多次访问主存,且指令长度比较长,所以M-M型指令在通用计算机中并不适宜。事实上,参与运算的两个操作数并不一定都在主存中,往往有一个或两个在通用寄存器中,此时的二地址指令就是R-M或R-R型指令,表3-1列出不同类型二地址指令的区别。

表3-1 不同类型二地址指令的区别

二地址指令类型	名　　称	操作数物理位置	执行速度	访问主存次数(取指除外)
M-M	存储器—存储器	主存	最慢	多次
R-R	寄存器—寄存器	寄存器	最快	不访问
R-M	寄存器—存储器	寄存器—主存	两者之间	一次

4. 指令系统中操作数的位置

按照CPU中操作数的存储位置,指令系统可分为堆栈型、累加器型和通用寄存器型3类,其相应的机器分别称为堆栈型机器、累加器型机器和通用寄存器型机器。注意,CPU中操作数的存储位置是针对指令系统中运算类指令来分类的。在堆栈型机器中,运算指令的操作数地址是隐含的,操作数在栈顶中,即前面所描述的零地址指令;在累加器型运算指

令中有一个操作数地址是隐含的，这个操作数在累加器中，即前面所描述的一地址指令。在通用寄存器型运算指令中，操作数全部是显式给出的，或者为寄存器地址，或者为主存地址，也就是前面所描述的二地址指令或三地址指令。

堆栈型机器的主要优点是表达式计算简单，指令短，但是堆栈不能被随机访问，并且栈顶容易形成瓶颈。累加器型机器的主要优点是机器的内部状态简单，指令短，实现容易，但是累加器同样容易成为计算机的瓶颈。通用寄存器型机器，编译器可以有效地计算表达式的值，可以减少访存次数，提高程序执行速度，但是指令中操作数需要显式给出，导致指令字较长。

通用寄存器型机器根据运算指令中存储器操作数的个数，又可以进一步分为3种类型：寄存器—寄存器(R-R)型，寄存器—存储器(R-M)型以及存储器—存储器(M-M)型。从表3-1中可以看出，R-R型指令系统运算类指令中不包含存储器操作数，因此运算速度快，目前的RISC机器均为这种指令系统。R-M型指令系统运算类指令中既有寄存器操作数又有存储器操作数，运算速度居中。而对于M-M型指令系统，由于要多次访存，运算速度最慢。当然，M-M型指令系统并不是所有运算类指令中的操作数均存放在存储器中，因为这与该指令系统仍然属于通用寄存器型指令系统是违背的。M-M型指令系统中只有部分运算指令的操作数都存放在存储器中。目前的指令系统中，通常只允许指令中最多具有一个存储器操作数。

5. 扩展操作码法

指令操作码的编码可以分为规整型和非规整型两类编码方式，最常用的非规整型编码方式是扩展操作码法。因为如果指令长度一定，则地址码与操作码字段的长度是相互制约的。为了解决这一矛盾，让操作数地址个数多的指令(三地址指令)的操作码字段短些，操作数地址个数少的指令(一或零地址指令)的操作码字段长些，这样既能充分地利用指令的各个字段，又能在不增加指令长度的情况下扩展操作码的位数，使它能表示更多的指令。假设某机的指令长度为16位，操作码字段为4位，有3个4位的地址码字段，如果按照定长编码的方法，4位操作码最多只能表示16条不同的三地址指令。

利用扩展操作码法可以在指令长度不变的情况下，使指令的总数远远大于16条。例如，指令系统中要求有15条三地址指令、15条二地址指令、15条一地址指令和16条零地址指令，共61条指令。扩展的方法如图3-1所示。

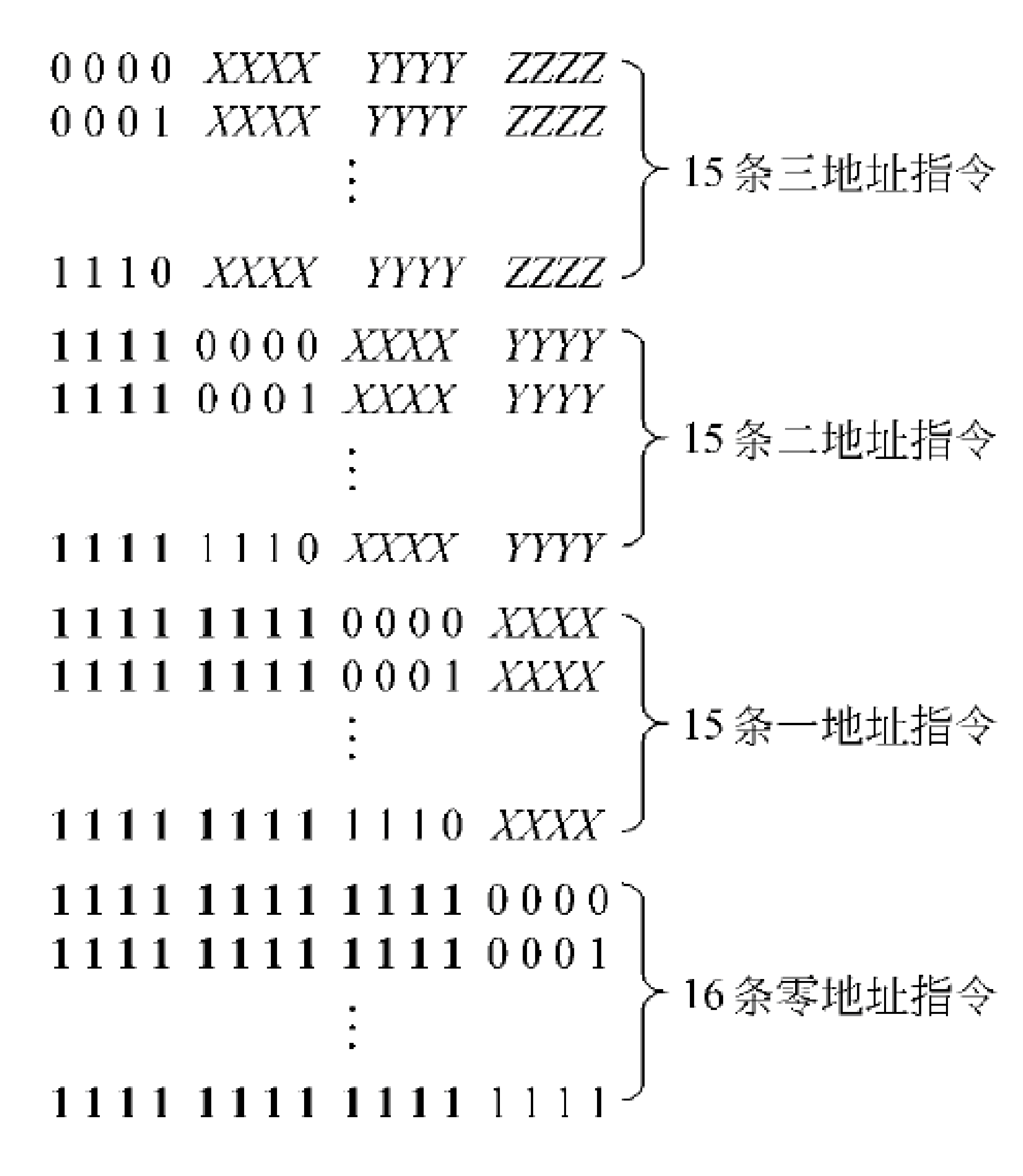


图3-1 扩展操作码举例

在上述扩展过程中，每种类型的指令都只留下一种编码作为扩展窗口。实际上可以有许多种不同的扩展方案。例如，也可以形成15条三地址指令、14条二地址指令、31条一地址指令和16条零地址指令，共76条指令，即：

(1) 4位操作码的编码0000～1110定义15条三地址指令，留下1111作为扩展窗口，与下一个4位组成一个8位的操作码字段；

(2) 8位操作码的编码11110000～11111101定义14条二地址指令，留下11111110和11111111作为扩展窗口，与下一个4位组成一个12位的操作码字段；

(3) 12位操作码的编码111111100000～111111111110定义31条一地址指令，扩展窗口为111111111111，与下一个4位组成16位的操作码字段；

(4) 最后，由16位操作码的编码1111111111110000～1111111111111111定义16条零地址指令。

不论采用何种方案，必须要注意以下两点：

- 不允许短码是长码的前缀，即短操作码不能与长操作码的前面部分的代码相同，否则将无法保证解码的唯一性和实时性。
- 各条指令的操作码一定不能重复相同，而且各类指令的格式安排应统一规整。

6. 字编址和字节编址

字编址是实现起来最容易的一种编址方式，这是因为每个编址单位与访问单位相一致，即每个编址单位所包含的信息量(二进制位数)与访问一次寄存器、主存所获得的信息量相同。早期的大多数机器都采用这种编址方式。

在采用字编址的机器中，每执行一条指令，程序计数器PC加1；每从主存中读出一个数据，地址寄存器加1。这种控制方式实现起来简单，地址信息没有任何浪费，其主要缺点是不支持非数值应用。

目前使用最普遍的编址方式是字节编址，这是为了适应非数值应用的需要。字节编址方式使编址单位与信息的基本单位(一个字节)相一致，但主存的访问单位是编址单位的若干倍。

在采用字节编址的机器中，如果指令长度是32位，那么每执行完一条指令，程序计数器要加4。如果数据字长是32位，当连续访问存储器时，每读写完一个数据字，地址寄存器要加4。由此可见，字节编址方式存在着地址信息的浪费。

7. 地址码位数与主存容量和最小寻址单位的关系

指令格式中每个地址码的位数是与主存容量和最小寻址单位(即编址单位)相关联的。主存容量越大，访问全部存储空间所需的地址码位数就越长。对于相同的存储容量来说，如果以字节为最小寻址单位，所需的地址码的位数就需要长些，但是可以方便地对每一个字符进行处理；如果以字为最小寻址单位(假定字长为16位或更长)，所需的地址码的位数可以减少，但对字符操作比较困难。例如，设某机主存容量为64MB，机器字长32b(位)，若最小寻址单位为字节(按字节编址)，其地址码应为26位($2^{26}=64MB$)；若最小寻址单位为字(按字编址)，其地址码只需24位。这是因为32位的一个字(W)等于4个字节(B)，则有：

$$64MB = \frac{64}{4}MW = \frac{2^{26}}{2^{2}}W = 2^{24}W = 16MW$$

从减少指令长度的角度看，最小寻址单位越大越好；而从对字符或位的操作是否方便的角度看，最小寻址单位越小越好。

8. 指令寻址和数据寻址

寻址可以分为指令寻址和数据寻址。寻找下一条将要执行的指令地址称为指令寻址，寻找操作数的地址称为数据寻址。

指令寻址比较简单，它又可以细分为顺序寻址和跳跃寻址。顺序寻址可通过程序计数

器(PC)加1,自动形成下一条指令的地址。需要说明的是,这里所说的PC加1中的1并不是物理上的数字1,而是逻辑上的数字1,即表示PC将指向下一条待执行的指令地址,具体PC值加多少取决于指令的长度和编址方式。跳跃寻址则需要通过程序转移类指令实现,其转移地址的形成方式有3种:直接(绝对)寻址、相对寻址和间接寻址。

数据寻址方式是根据指令中给出的地址码字段寻找真实操作数地址的方式。数据寻址的种类较多,如立即寻址、寄存器寻址、直接寻址、间接寻址、变址寻址等,其最终目的都是寻找所需要的操作数。

9. 各种数据寻址方式的速度区别

数据寻址的最终目的是寻找所需要的操作数。操作数可以在主存中,也可以在寄存器中,甚至可以在堆栈中。各种不同的寻址方式获取操作数的速度是不相同的,其中速度最快的是立即寻址,最慢的是多级间接寻址。

立即寻址是一种特殊的寻址方式,指令中在操作码字段后面的部分不是通常意义上的地址码,而是操作数本身,也就是数据就包含在指令中,只要取出指令,也就取出了可以立即使用的操作数,不必再次访问存储器,从而提高了指令的执行速度。

寄存器寻址获取操作数的速度仅次于立即寻址,因为操作数在通用寄存器中,所以不需要访问主存就可以获得数据。

直接寻址、寄存器间接寻址、变址寻址、基址寻址、相对寻址和页面寻址等寻址方式获取一个操作数都只需要访问一次主存,根据有效地址EA得到的难易程度,速度由快至慢依次为:直接寻址、寄存器间接寻址、页面寻址、变址寻址(基址寻址、相对寻址)。

间接寻址指令中给出的形式地址A不是操作数的地址而是操作数地址的地址。这就意味着为获取一个操作数,至少需要两次访问主存,第一次得到操作数的有效地址,第二次才能得到操作数。间接寻址允许多级间址,多级间接寻址为获取操作数需要多次访问主存,即使在找到操作数有效地址后,还须再访问一次主存才可得到真正的操作数。

基本寻址方式的比较如表3-2所示。表中列出的偏移寻址包括变址寻址、基址寻址和相对寻址3种方式。

表3-2 基本寻址方式的比较

寻址方式	规则	主要优点	主要缺点
立即寻址	操作数=A	无须访问存储器	操作数范围受限
寄存器寻址	EA=R	无须访问存储器	寻址空间受限
直接寻址	EA=A	简单	寻址空间受限
间接寻址	EA=(A)	寻址空间大	多次访问主存
寄存器间接寻址	EA=(R)	寻址空间大	多访问一次主存
偏移寻址	EA=(R)+A	灵活	复杂

10. 寄存器寻址的特点

寄存器寻址指令在执行过程中所需要的操作数来源于寄存器,运算结果也写回到寄存器中,这种寻址方式在所有的RISC计算机以及大部分CISC计算机中得到广泛应用。

寄存器寻址方式的优点主要有:

(1) 指令字长短。由于通用寄存器的数量一般只有几十个,在指令中只需很少几位就能表示一个操作数的地址。

(2) 指令执行速度快。由于访问寄存器的速度很快,与主存相比,访问时间几乎可以忽略不计。

(3) 支持向量、矩阵运算。当通用寄存器的数量比较多时,可以把一个向量或向量的一部分放在通用寄存器内,从而提高运算速度。

寄存器寻址方式也有明显的缺点,主要有:

(1) 不利于优化编译。由于通用寄存器分配是否合理将直接影响到程序的执行速度,所以通常要把那些连续使用或用得比较频繁的变量分配在通用寄存器中,这就给编译器的优化设计造成了很大的困难。

(2) 现场切换困难。在程序运行过程中,当发生调用、中断、分时切换等情况时,要把有关通用寄存器中的内容都保存到主存中,在程序返回时,再全部恢复这些通用寄存器中的内容。通用寄存器的数量越多,保存和恢复所需要的时间就越长。为了解决这一问题,目前多数处理机都设置有两套或两套以上的通用寄存器,程序员只能看到其中一套通用寄存器,当发生现场切换时,硬件自动切换到另一套通用寄存器。

(3) 硬件复杂。一方面,在处理机中设置大量的通用寄存器,包括程序员看见的寄存器和程序员看不见的寄存器,需要增加硬件;另一方面,这些寄存器的读、写及现场切换等控制也相当复杂。

11. 直接寻址与寄存器间接寻址

直接寻址与寄存器间接寻址都属于主存寻址,这是所有计算机中都普遍采用的一类寻址方式。

直接寻址又称为绝对寻址,它是在指令中直接给出、参加运算的操作数及运算结果所存放的主存地址,即在指令中直接给出有效地址。在早期生产的计算机及目前某些专用计算机中用得比较多。随着主存容量的不断扩大及虚拟存储器的普及,这种寻址方式也暴露出了许多致命的弱点。首先,它需要很长的地址码,特别在二地址及三地址指令中,这一矛盾更为突出。其次,为了实现程序循环及高效处理数组运算等,程序设计中通常需要修改数据的地址,而采用直接寻址方式编写的程序,如果要修改数据地址就必须修改程序中的指令本身,采用这种方法编写的程序将没有再入性,这是现代程序设计思想所不能接受的。

寄存器间接寻址指令所指定的寄存器中存放着操作数的有效地址,而操作数则存放在主存中。这种方式既利用了寄存器寻址方式指令字长短、执行速度快的优点,又避免了直接寻址方式修改数据地址就必须修改指令本身的弱点。

12. 间接寻址与变址寻址

对于数组运算,通常要用一个循环程序对数组中的各个元素进行操作,这时必须通过修改操作数的地址才能实现。间接寻址方式与变址寻址方式的设计目标都是为了解决操作数地址的修改问题。它们都可以在程序设计过程中对操作数的地址进行修改,而不必去修改程序中的指令本身。

原则上,在一个计算机系统中,仅需设置间接寻址与变址寻址方式中的任何一种即可,如在IBM公司生产的大型、中型计算机系统中,只有变址寻址方式,没有间接寻址方式。在一些小型及微型计算机系统中,只有间接寻址方式,没有变址寻址方式。也有一些计算机系

统，间接寻址方式和变址寻址方式两种都有。

这两种寻址方式的区别如下：

(1) 实现的难易程度。间接寻址实现起来很容易，只需要增加一条从主存的数据寄存器到地址寄存器的数据通路即可。而变址寻址的实现则需要增加较多的硬件，如需要一个加法器，一个和多个变址寄存器(也可以与通用寄存器合用)。

(2) 指令的执行速度。采用间接寻址方式编写的程序，执行速度较慢。读写一个操作数至少需要访问两次主存，第一次读取有效地址，第二次才是读或写操作数，如果采用多级间接寻址，则访问主存的次数还有增多。而采用变址寻址方式编写的程序，执行速度较快。有效地址通过加法器直接产生，不需要访问主存。

(3) 对数组运算的支持。变址寻址方式比较好，间接寻址方式比较差，这是因为变址寻址方式可以修改变址值。

13. 变址寻址与基址寻址

变址寻址是将变址寄存器 R_x 的内容与指令的形式地址 A 相加，形成操作数有效地址，即 $EA=(R_x)+A$。R_x 的内容称为变址值。

基址寻址是将基址寄存器 R_b 的内容与指令的形式地址(位移量)相加，形成操作数有效地址，考虑到位移量实际上是一个可正、可负的带符号数，故用 D 表示，即 $EA=(R_b)+D$。R_b 的内容称为基址值。

基址寻址和变址寻址在形成有效地址时所用的算法相同，但使用它们的目的却有所不同。例如，在数组运算中使用时，通常变址寄存器的内容是可变的，而基址寄存器的内容却保持不变。如果需要对一个数组中的每一个元素或者一部分连续存放的元素进行操作时，则可将该数组的首地址作为指令中的形式地址，而将需要操作的第一个元素的序号置入变址寄存器中，于是这条指令执行一次可得到数组中的一个元素，然后将变址寄存器内容加1，同一条指令的下一次执行就可以得到数组中的下一个元素，重复上述操作，可以顺序取得数组中的全部和部分元素。如果需要从一个数组中找到某一个元素，则可将该数组的首地址置入基址寄存器中，所需元素的序号作为指令中的形式地址，于是可以采用基址寻址方式方便地获取所需的元素。

在有些计算机中还设置有基址加变址寻址方式，若将数组的首地址置入基址寄存器，将数组中元素的序号置入变址寄存器，便可方便地实现对该数组中的多个元素进行某种操作。

14. 偏移寻址中的相对寻址

将直接寻址和寄存器间接寻址方式相结合，可以得到几种寻址方式。因为它们提供操作数有效地址的机制相同，都是将指定寄存器的内容与指令中的地址码字段相加，所以统称为偏移寻址。常见的偏移寻址有相对寻址和上述的变址寻址、基址寻址 3 种。

相对寻址是基址寻址的一种变通，由程序计数器 PC 提供基准地址，即 $EA=(PC)+D$。

由于大多数访存的位置都相对靠近正在执行的指令位置，则使用相对寻址可节省指令中的地址码位数，而且采用相对寻址方式编制程序，不需要指定绝对地址，只需确定程序内部的相对距离，因而可以使用浮动地址，这样给编程带来了方便。

15. 页面寻址

页面寻址相当于将整个主存空间分成若干个大小相同的区，每个区称为一页，每页有若干个主存单元。每页都有自己的编号，称为页面地址；页面内的每个主存单元也有自己的编

号,称为页内地址。这样,存储器的有效地址就被分为两部分:前部为页面地址,后部为页内地址。页面地址的位数取决于主存划分页数的多少,页内地址的位数取决于每一页中存储单元的多少。页面寻址根据页面地址的形成方式又可以分成3种形式:

(1) 基页寻址,又称零页寻址。由于页面地址全等于0,所以有效地址EA=0//A(//在这里表示简单拼接),操作数S一定在零页面中。基页寻址实际上就是直接寻址。

(2) 当前页寻址。页面地址就等于程序计数器PC的高位部分的内容,所以有效地址EA=$(PC)_H$//A,操作数S与指令本身一定处于同一页面中。$(PC)_H$代表程序计数器高位部分的内容,当前页寻址示意图参见图3-2。

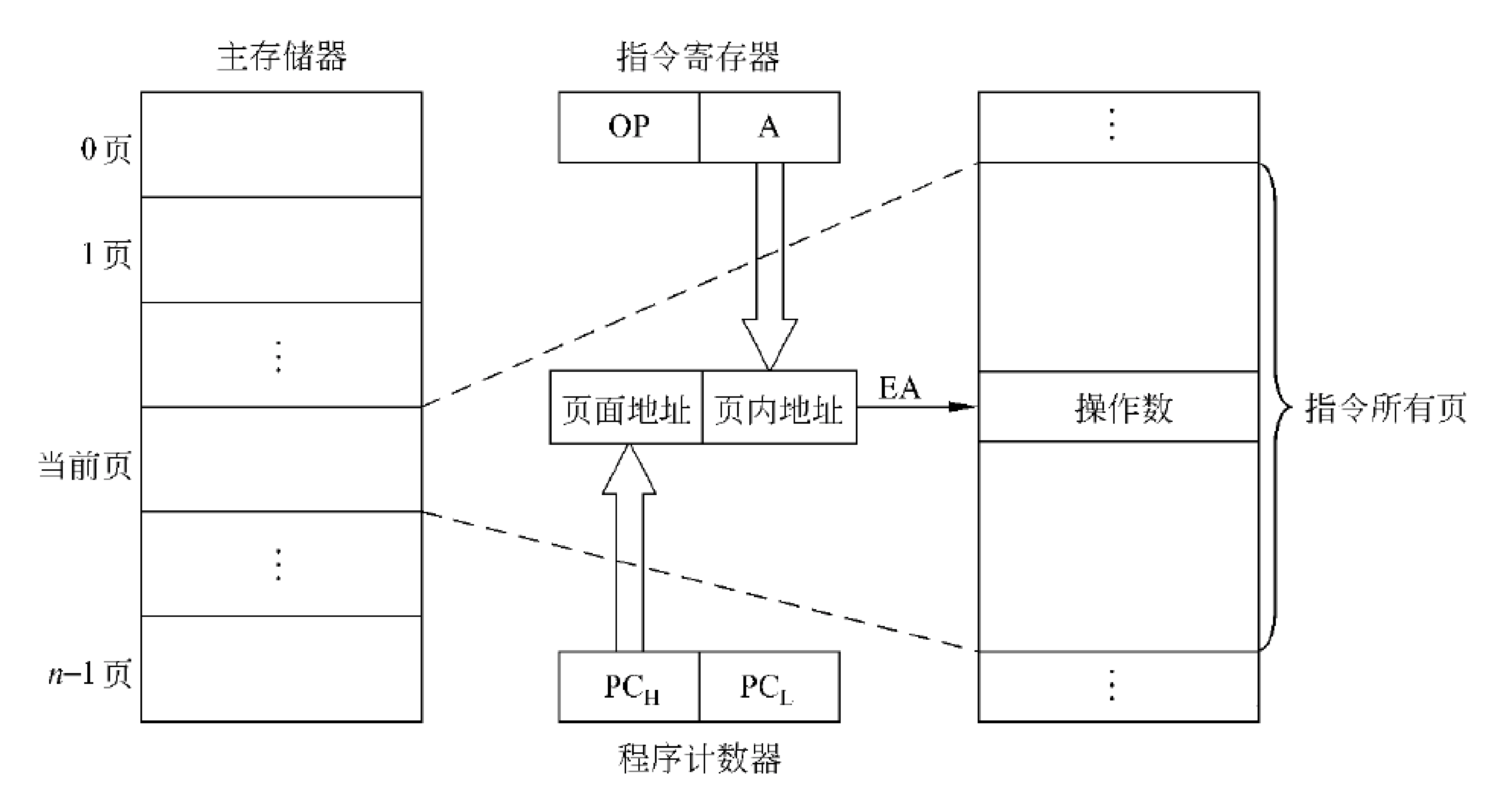


图3-2　当前页寻址示意图

(3) 页寄存器寻址。页面地址取自页寄存器,与形式地址相拼接形成有效地址。

通常所说的页面寻址主要是指当前页寻址。

在页面寻址方式时,页内地址由指令的地址码部分自动直接提供,它与页面地址通过简单的拼装连接就可得到有效地址,无须像偏移寻址那样进行加法计算,所以可以迅速快捷地得到有效地址。

16. 堆栈寻址

大多数计算机都设置有堆栈,寻找堆栈中的数据称为堆栈寻址方式。堆栈寻址方式的地址是隐含的,在指令中不必给出操作数的地址。参加运算所需要的操作数从堆栈顶端弹出,如果需要两个或多个操作数,则依次从堆栈顶端弹出,运算结果压入堆栈顶端。

自20世纪60年代开始,出现了一批以堆栈寻址方式为主的堆栈计算机,堆栈计算机具有如下特点:

(1) 支持高级语言,有利于编译程序。因为一般的算术表达式可以容易地转化成逆波兰表达式(后缀表达式),而逆波兰表达式能够直接形成由堆栈指令组成的程序,这样就简化了编译程序。

(2) 程序的总存储量最短。由于堆栈指令不需要地址码,指令的长度很短,与以寄存器寻址方式和以主存寻址方式的计算机系统相比,虽然程序本身的条数没有减少,但程序的总存储量缩短许多。

(3) 支持程序的嵌套和递归调用，支持中断处理。在程序调用过程中，要保存返回地址，保存程序现场，并向子程序传送参数。在堆栈计算机中，可以把这些信息都压入堆栈。当从子程序返回时，可以直接从堆栈中弹出所需要的信息。这样，可以减少大量的辅助操作，加快运算速度。

堆栈计算机主要缺点是运算速度比较慢，这是由于堆栈与处理机之间的信息传送量大造成的。

17. 存储器堆栈栈指针的修改

从主存中划出一段区域来作堆栈是最合算且最常用的方法，这种堆栈又称为软堆栈。堆栈的大小可变、栈底固定、栈顶浮动，需要一个专门的硬件寄存器作为堆栈栈顶指针 SP，栈指针所指定的存储单元就是堆栈的栈顶，当堆栈中的数据发生变化时，栈指针会改变。

对于自底向上生成的堆栈，其栈底地址大于栈顶地址，如果栈指针始终指向栈顶的满单元，则进栈和出栈的两种操作如下。

进栈时，堆栈指针 SP 的内容需要先自动减量，然后再将数据压入堆栈，例如：

(SP) − 1→SP	修改栈指针
(A)→ (SP)	将 A 中的内容压入栈顶单元

出栈时，需要先将堆栈中的数据弹出，然后 SP 的内容再自动增量，例如：

((SP))→A	将栈顶单元内容弹出送入 A 中
(SP) + 1→SP	修改栈指针

在上述两种操作中，A 为寄存器或存储单元地址；(SP)表示堆栈指针的内容，即栈顶单元地址；((SP))表示栈顶单元的内容。式中的 1 代表栈指针增加或减少的量，它是由压入堆栈的数据字长决定的，若存储器按字节编址，压入堆栈的数据为 32 位，则修改栈指针应该是(SP)－4→SP。

对于自顶向下生成的堆栈(栈底地址小于栈顶地址)，修改栈指针时的增、减量正好与上述操作相反。如果栈指针始终指向栈顶的空单元，则进栈和出栈相应的两种操作的顺序也要颠倒过来。

堆栈操作既不是在堆栈中移动它所存储的内容，也不是把已存储在栈中的内容从栈中抹掉，而是通过调整堆栈指针来给出新的栈顶位置，以便对位于栈顶位置的数据进行操作。

18. 程序控制类指令的特点

程序控制类指令用于控制程序的执行顺序，主要包括转移指令、子程序调用指令和返回指令等。

转移指令又分为无条件转移指令和条件转移指令两种。无论是条件转移还是无条件转移都需要给出转移地址。若采用相对寻址方式，转移地址为当前指令地址(即 PC 的值)和指令中给出的位移量之和，即(PC)＋位移量→PC；若采用绝对寻址方式，转移地址由指令的地址码字段直接给出，即 A→PC。

子程序是一组可以公用的指令序列，只要知道子程序的入口地址就能调用它。子程序调用(转子)指令安排在主程序中需要调用子程序的地方，转子指令需要给出转移的目的地址(子程序的入口地址)，所以它必定是一地址指令。

转移指令和转子指令都可以改变程序的执行顺序，但存在的主要差别有：

(1) 转移指令使程序转移到新的地址后继续执行指令,不存在返回的问题,所以没有返回地址;而转子指令要考虑返回问题,所以必须以某种方式保存返回地址(转子指令的下一条指令的地址),以便返回时能找到原来的位置。

(2) 转移指令用于实现同一个程序内的转移;而转子指令转去执行一段子程序,实现的是程序段与程序段之间的转移。

子程序可以再调用别的子程序,这称为子程序的嵌套调用,子程序还可以自己调用自己,这称为子程序的递归调用。直接自己调用自己的递归调用称为直接递归调用,间接自己调用自己的递归调用称为间接递归调用。

从子程序转向主程序的指令称为返回指令,子程序的最后一条指令一定是返回指令。返回地址存放的位置决定了返回指令的格式,返回地址通常保存在堆栈中,所以返回指令常是零地址指令。对于没有堆栈的计算机,返回地址也可以保存在其他地方。例如,用子程序的第一个字单元来保存返回地址。转子指令把返回地址存放在子程序的第一个字单元中,子程序从第二个字单元开始执行。返回时将第一个字单元地址作为间接地址,采用间址寻址方式返回主程序,此时的返回指令必须是一条一地址指令。

19. 条件转移指令的转移条件

绝大多数算术运算指令都会影响到状态标志位,通常的标志位有进位/借位标志(CF)、零标志(ZF)、符号标志(SF)、溢出标志(OF)和奇偶标志(PF)等。

运算类指令除常见的加、减、乘、除指令外,还包括比较指令。比较指令(CMP)与减法指令(SUB)都执行减法操作,但前者不保留运算结果,只是改变状态标志位,而后者不仅要保留运算结果,也要改变标志位。

表3-3给出了在无符号数比较和带符号数比较两种情况下,其状态标志反映的两数大小关系。从表中可以看出,对无符号数和带符号数,根据标志位状态来判断两数大小的条件是不同的:前者依据CF和ZF进行判断,后者则依据ZF、SF和OF进行判断。例如,要判断 $A<B$ 成立,无符号数所用的条件是 ZF=0,CF=1,而带符号数所用的条件是 ZF=0,OF+SF=1。

表3-3 状态标志反映的两数关系

两数比较结果(A-B)		受影响标志			
		CF	ZF	SF	OF
$A=B$(Equal)		0	1	0	0
无符号数	$A<B$(Below)	1	0	—	—
	$A>B$(Above)	0	0	—	—
带符号数	$A<B$(Less)	—	0	1	0
		—	0	0	1
	$A>B$(Greater)	—	0	1	1
		—	0	0	0

CMP指令常用于比较两个数,后将跟条件转移指令进行程序控制转移。条件转移必须

受到条件的约束,若条件满足时才执行转移,否则程序仍顺序执行。但需要注意,正因为判断无符号数和带符号数大小的条件不同,所以条件转移指令也分无符号数和带符号数两类不同的条件转移指令,见表 3-4。

表 3-4　条件转移指令

	转移条件	含　义
无符号数条件转移指令	CF=1	有进位转移(与低于/不高于等于转移重叠)
	CF=0	无进位转移(与高于或等于/不低于转移重叠)
	PF=1	奇偶位为 1 转移
	PF=0	奇偶位为 0 转移
	CF=ZF=0	高于/不低于等于转移
	CF=0	高于或等于/不低于转移
	CF=1	低于/不高于等于转移
	CF=1 或 ZF=1	低于或等于/不高于转移
	ZF=1	等于/为零转移
	ZF=0	不等于/非零转移
带符号数条件转移指令	OF=1	溢出转移
	OF=0	无溢出转移
	SF=1	为负数转移
	SF=0	为正数转移
	ZF=0 且 SF=OF	大于/不小于等于转移
	SF=OF	大于或等于/不小于转移
	SF≠OF	小于/不大于等于转移
	ZF=1 或 SF≠OF	小于或等于/不大于转移
	(CX)=0	CX 寄存器为 0 转移

20. 输入输出指令的特点

输入输出指令用来实现主机与外部设备之间的信息交换,包括输入输出数据、主机向外设发控制命令或外设向主机报告工作状态等。各种不同计算机的 I/O 指令差别很大,通常有下面两种方式。

1) 独立编址

外设寄存器和主存单元分别独立编址。在这种方式下,使用专门的输入(IN)/输出(OUT)指令,指令中必须给出外部设备的编号(端口地址)。

2) 统一编址

外设寄存器和主存单元统一编址。在这种方式下,不需要专门的 I/O 指令,就用一般的数据传送类指令来实现 I/O 操作。一个外部设备通常至少有两个寄存器(数据寄存器和命令/状态寄存器),每个外设寄存器都有唯一的地址,CPU 可以像访问主存一样去访问外

部设备的寄存器。

3.3 典型例题详解

【例 3.1】 试比较寄存器—寄存器型指令与存储器—存储器型指令的优缺点。

解：在寄存器—寄存器(R-R)型指令系统中，运算类指令中不含存储器型操作数。机器在执行这类指令的过程中，只对寄存器中的操作数进行操作，从寄存器中取操作数，结果也放在寄存器中，不需要访问存储器，因此速度很快。但是，由于所有数据来源于存储器并且最终要存放到存储器中，故寄存器—寄存器型指令的执行需要使用访问存储器指令来从存储器中存取操作数。

在存储器—存储器(M-M)型指令系统中，部分运算类指令的操作数都存放在存储器中。两个以上的操作数存放在存储器中，使得指令的执行需要 4 次以上的访问存储器(1 次取指、2 次读操作数、1 次写操作数)，这导致指令执行速度缓慢。这种指令可以不使用寄存器，从这个角度看，CPU 的实现代价相对较低。

【例 3.2】 某机字长 32 位，指令单字长，指令系统中具有二地址指令、一地址指令和零地址指令各若干条，已知每个地址长 12 位，采用扩展操作码方式，问该指令系统中的二地址指令、一地址指令、零地址指令各最多能有多少条？

解：扩展操作码的指令字长是固定的，但指令中操作码的位数不固定。设指令字长 32 位，每个地址字段为 12 位，因此，零地址指令的操作码占 32 位，一地址指令的操作码占 20 位，二地址指令的操作码占 8 位。

对于二地址指令，至少需要留出一个扩展窗口给一地址指令。显然最多可以有 2^8-1 条二地址指令。

对于一地址指令，最多的情况是指令系统中只有一条二地址指令，并要为零地址指令留下一个扩展窗口。显然最多可以有 $(2^8-1)\times2^{12}-1$ 条一地址指令。

对于零地址指令，最多的情况是指令系统中只有一条二地址指令和一条一地址指令，其余均为零地址指令。零地址指令的操作码为 32 位，共有 2^{32} 种编码，其中 2^{24} 种编码用作表示某条二地址指令，2^{12} 种编码用作表示某条一地址指令。因此零地址指令最多可以有 $2^{32}-2^{24}-2^{12}$ 条。

【例 3.3】 指令字长为 12 位，每个地址码为 3 位，采用扩展操作码的方式，设计 4 条三地址指令、16 条二地址指令、64 条一地址指令和 16 条零地址指令。

(1) 给出一种操作码的扩展方案。

(2) 画出指令译码逻辑。

(3) 计算操作码的平均长度。

解：(1) 操作码的扩展方案如图 3-3 所示。

(2)指令译码逻辑如图 3-4 所示。

(3) 操作码的平均长度＝全部指令的操作码长度÷指令总数＝(4×3＋16×6＋64×9＋16×12)÷



图 3-3 操作码的扩展方案

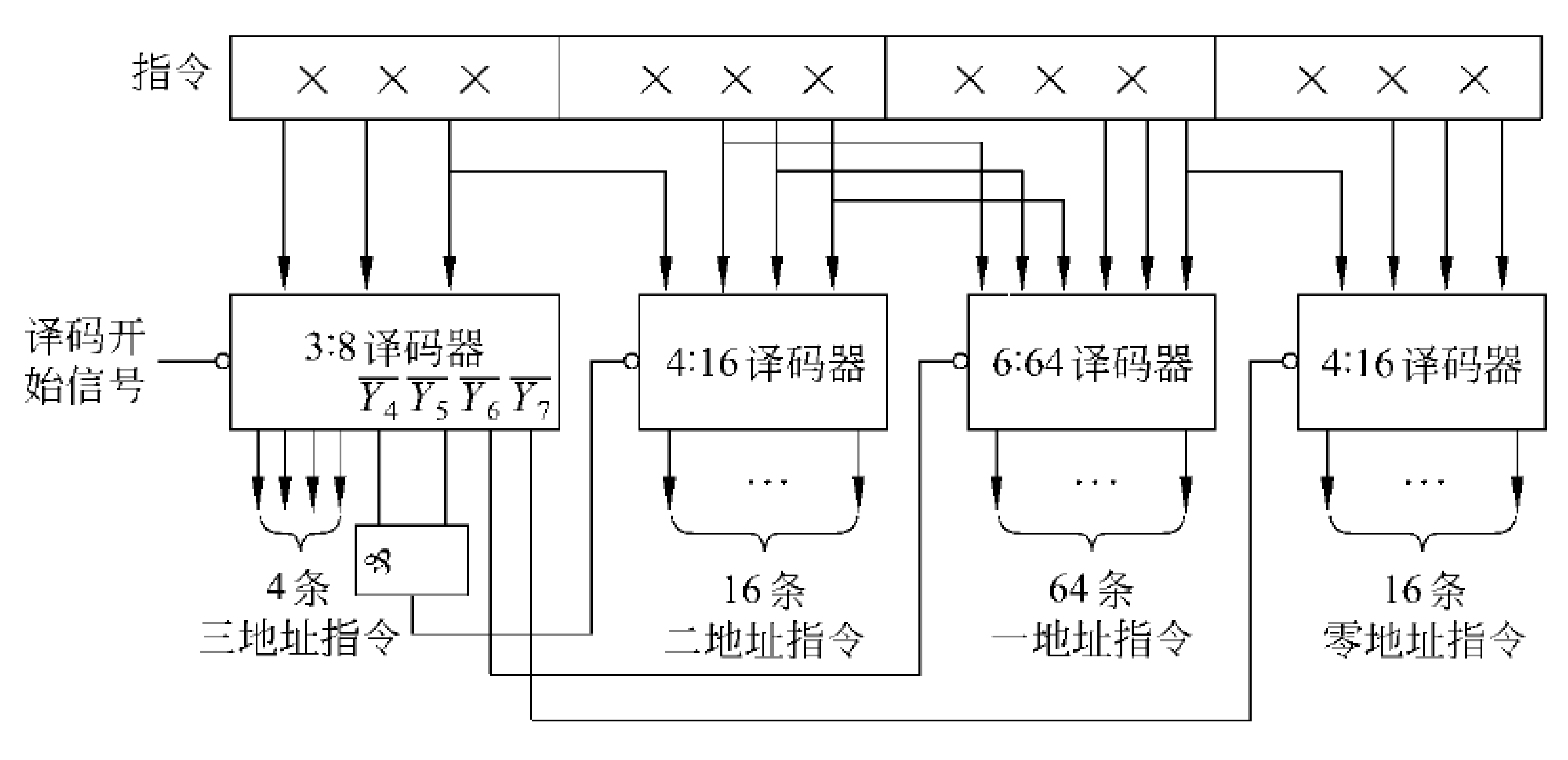


图 3-4　指令译码逻辑

(4+16+64+16)=8.7。

【例 3.4】 在 16 位长的指令系统中，设计一个扩展操作码，能对下列指令进行译码。

(1) 7 条三地址指令。

(2) 225 条单地址指令。

(3) 16 条零地址指令。

令每个地址码为 4 位，分别画出 3 种类型指令的格式，并说明译码过程。

解：(1) 7 条三地址指令：

$$0000\ XXXX\ YYYY\ ZZZZ$$
$$\vdots$$
$$0110\ XXXX\ YYYY\ ZZZZ$$

(2) 225 条单地址指令：

以扩展窗口为 1000 的单地址指令最多可以有 256 种不同的操作码编码，现在只有 225 条指令，多余的 31 种编码为非法操作码。

11111111	最后一个操作码编码
− 11111	31 种非法操作码
11100000	最后一条单地址指令的操作码

$$1000\ 0000\ 0000\ XXXX$$
$$\vdots$$
$$1000\ 1110\ 0000\ XXXX$$

(3) 16 条无地址指令：

$$1111\ 0000\ 0000\ 0000$$
$$\vdots$$
$$1111\ 0000\ 0000\ 1111$$

3 种类型指令的格式如图 3-5 所示。为了译码的方便，特别安排三地址指令的识别标志是指令的最高位为 0，接着再对剩余的 3 位操作码译码，可区别出 7 条不同的三地址指令。单地址和零地址指令的最高位都为 1，如果次高位为 0，则是单地址指令，接着将指令的中间 8 位作为操作码译码，可以区分出 225 条单地址指令；如果次高位仍为 1，则是零地址

指令,接着对指令的最低4位进行译码,可以区分出16条零地址指令。

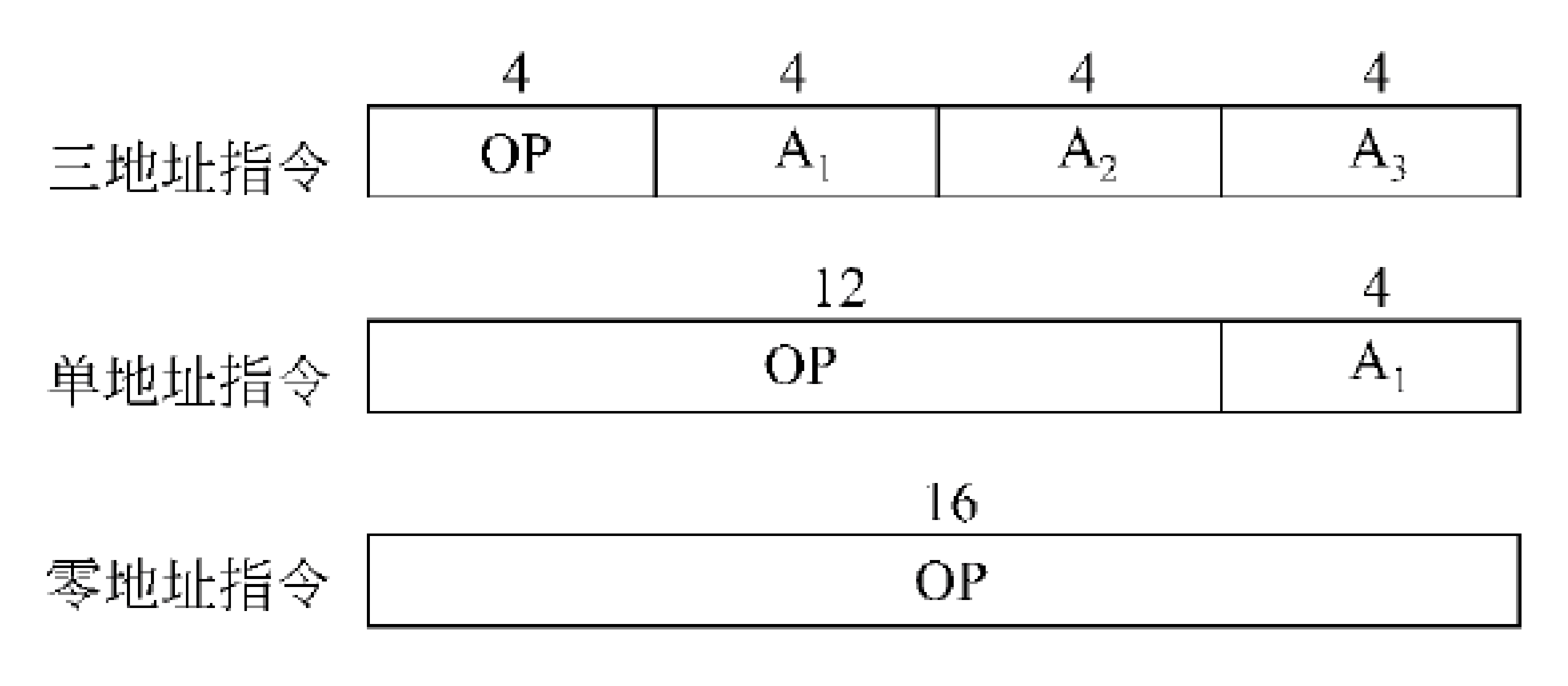


图3-5　3种类型指令的格式

【例3.5】 假定一个32位的微处理器,指令字长32位,每条指令由两部分组成,其中第一个字节为操作码,剩余的为立即数或操作数地址。

(1) 可直接访问的最大主存空间是多少?

(2) 讨论下列两种情况对系统速度的影响:微处理器总线使用32位局部地址总线和16位局部数据总线;微处理器总线使用16位局部地址总线和32位局部数据总线。

(3) 程序计数器和指令寄存器各需要多少位?

解:(1) 因为32位指令中,有8位为操作码,其余24位为操作数地址,所以可直接访问的最大主存空间为2^{24}。

(2) 若采用32位局部地址总线和16位局部数据总线,则需要两个访存周期才能读取一个字的指令和数据;若采用16位局部地址总线和32位局部数据总线,则需要两个时钟周期才能把地址送出去。以上两种情况都会导致系统速度下降。

(3) 指令寄存器与指令字长相同,应为32位。而程序计数器应与可访问的主存空间相对应,应为24位。

【例3.6】 某计算机的字长为16位,存储器按字编址,访存指令格式如图3-6所示。其中,OP是操作码,MOD定义寻址方式,见表3-5,A为形式地址。设PC和R_x分别为程序计数器和变址寄存器,字长为16位,问:

15　　　11	10　　8	7　　　　0
OP	MOD	A

图3-6　访存指令格式

(1) 该格式能定义多少种指令?

(2) 各种寻址方式的寻址范围为多少字?

(3) 写出各种寻址方式的有效地址EA的计算式。

解:(1) 由图3-6可知,指令格式中的高5位为操作码字段,总的指令种类为$2^5=32$种。

(2) 假设存储器按字编址,各种寻址方式的寻址范围见表3-6。

表3-5　寻址方式定义

M值	寻址方式
0	立即寻址
1	直接寻址
2	间接寻址
3	变址寻址
4	相对寻址

表3-6　寻址范围

寻址方式	寻址范围
立即寻址	指令字本身
直接寻址	256个字(主存中最前边的256个字)
间接寻址	64K个字
变址寻址	64K个字
相对寻址	256个字(PC值附近的256个字)

(3) 写出各种寻址方式的有效地址 EA 的计算式。

寻址方式	有效地址表达式
0	EA=(PC),即操作数在指令中(指令字的低位部分)
1	EA=A
2	EA=(A)
3	EA=(R_x)+A
4	EA=(PC)+A

【例 3.7】 某机字长 32 位,共有机器指令 100 条,指令单字长,等长操作码,CPU 内部有通用寄存器 32 个,可做变址寄存器用。存储器按字节编址,指令拟用直接寻址、间接寻址、变址寻址和相对寻址 4 种方式。

(1) 分别画出采用 4 种不同寻址方式的单地址指令的指令格式;

(2) 采用直接寻址和间接寻址方式时,可寻址的存储器空间各是多少?

(3) 写出 4 种方式下,有效地址 EA 的表达式。

解:(1) 由于系统共有 100 条指令,满足这个条件的最小指令操作码位数为 7 位;系统中允许使用 4 种不同的寻址方式,寻址方式字段需要 2 位。

指令长度 32 位,剩余位数=32-7-2=23 位,即为地址码字段。但在变址寻址时,还需要有 5 位寄存器编码,所以真正的地址码只有 18 位。4 种不同寻址方式的单地址指令的指令格式如图 3-7 所示。

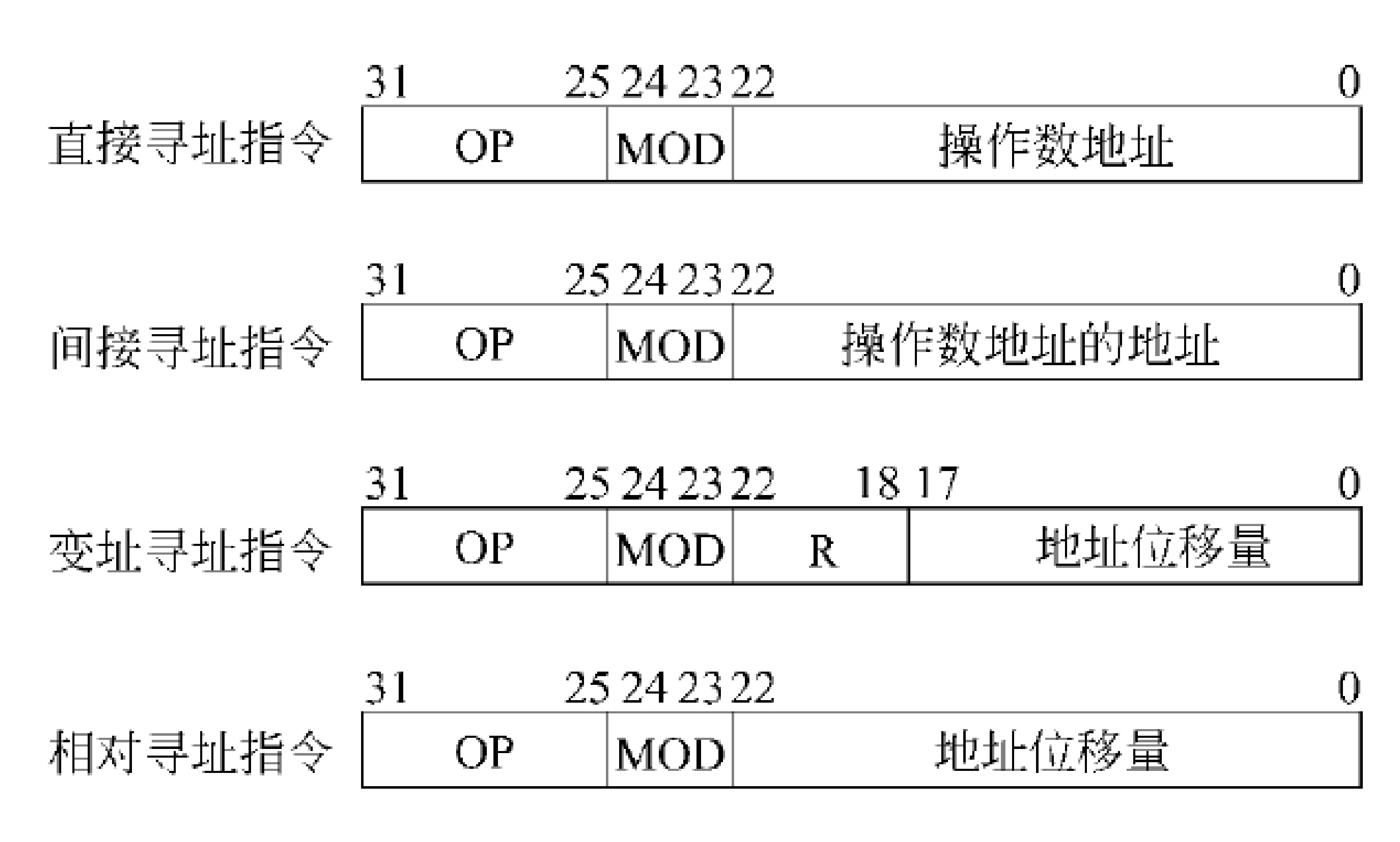


图 3-7 单地址指令的指令格式

(2) 存储器按字节编址,采用直接寻址时,寻址范围为 8MB(2^{23});采用间接寻址时,由于机器的字长为 32 位,所以可寻址范围为 4GB(2^{32})。

(3) 4 种寻址方式下,有效地址 EA 的表达式为:

直接寻址	EA=A
间接寻址	EA=(A)
变址寻址	EA=(R_x)+A
相对寻址	EA=(PC)+A

【例 3.8】 某机字长为 16 位,采用一地址格式的指令系统,允许直接、间接、变址、基址寻址,变址寄存器和基址寄存器均为 16 位,试回答:

(1) 若采用单字指令,共能完成 108 种操作,画出指令格式,并指出直接寻址和一次间

址的寻址范围各为多少?

(2) 若采用双字指令,操作码位数和寻址方式不变,指令可直接寻址的范围又是多少?画出指令格式。

(3) 字长不变,可采用什么方法访问容量为 8MB 的主存任一地址单元?说明理由。

解:(1) 由于系统的指令集有 108 条指令,满足这个条件的最小指令操作码位数为 7 位;系统中允许有 4 种不同的寻址方式,寻址方式字段需要 2 位。故采用单字指令时,地址码的位数为 7 位。设存储器按字编址,直接寻址的范围为 2^7 个字。采用一次间址寻址时,由于地址码字段中是操作数地址的地址,操作数的有效地址在存储器内,该地址是 16 位的,因此,间址寻址的范围为 2^{16}个字。指令格式如图 3-8(a)所示。

(2) 当采用双字指令时,指令中地址码字段的长度增加,所以访问的范围也加大。该范围为 $2^{(16+7)}=2^{23}$(8M)个字。指令格式如图 3-8 (b)所示。

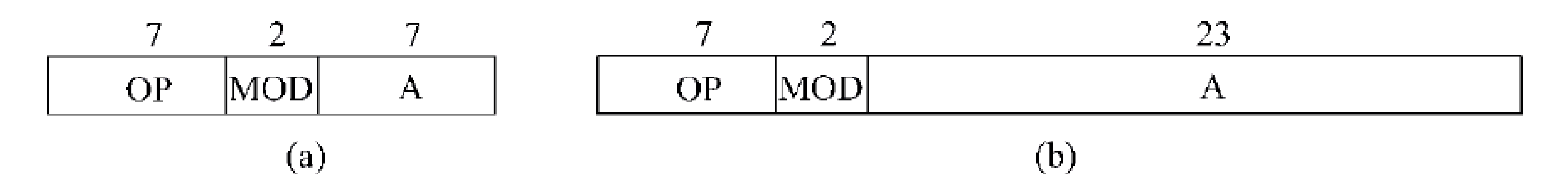


图 3-8 某机的指令格式

(3) 采用双字指令的直接寻址方式即可以访问 8MW,但这样每条指令需要占用两个存储字,处理上比较复杂且代价高。也可以使用变址或基址寻址来访问比较大的寻址空间,因为 8MB=4MW,该存储器按字编址,物理空间需要 22 位地址。由于变址或基址寄存器只有 16 位,变址或基址寻址的有效地址 EA=(R)+A,可以将变址或基址寄存器中的内容左移 6 位之后再与 A 相加。

【例 3.9】 设某计算机有变址寻址、间接寻址和相对寻址等寻址方式,设当前指令的地址码部分为 001AH,正在执行的指令所在地址为 1F05H,变址寄存器中的内容为 23A0H。

(1) 当执行取数指令时,如为变址寻址方式,则取出的数为多少?

(2) 如为间接寻址,取出的数为多少?

(3) 当执行转移指令时,转移地址为多少?

已知存储器的部分地址及相应内容,见表 3-7。

表 3-7 存储单元地址与内容

地　址	内　容
001AH	23A0H
1F05H	241AH
1F1FH	2500H
23A0H	2600H
23BAH	1748H

解:(1) 这是一种数据寻址(变址寻址),操作数 $S=((R_x)+A)=(23A0H+001AH)=(23BAH)=1748H$。

(2) 这也是一种数据寻址(间接寻址),操作数 $S=((A))=((001AH))=(23A0H)=2600H$。

(3) 这是一种指令寻址,转移指令使用相对寻址,转移地址$=(PC)+A=1F05H+001AH=1F1FH$。

因为在本题中没有指出指令的长度,故此题未考虑 PC 值的更新。

【例 3.10】 一条双字长的 LOAD 指令存储在地址为 200 和 201 的存储位置,该指令将指定的内容装入累加器(AC)中。指令的第一个字指定操作码和寻址方式,第二个字是地址

部分。寄存器与主存内容示意图如图 3-9 所示。

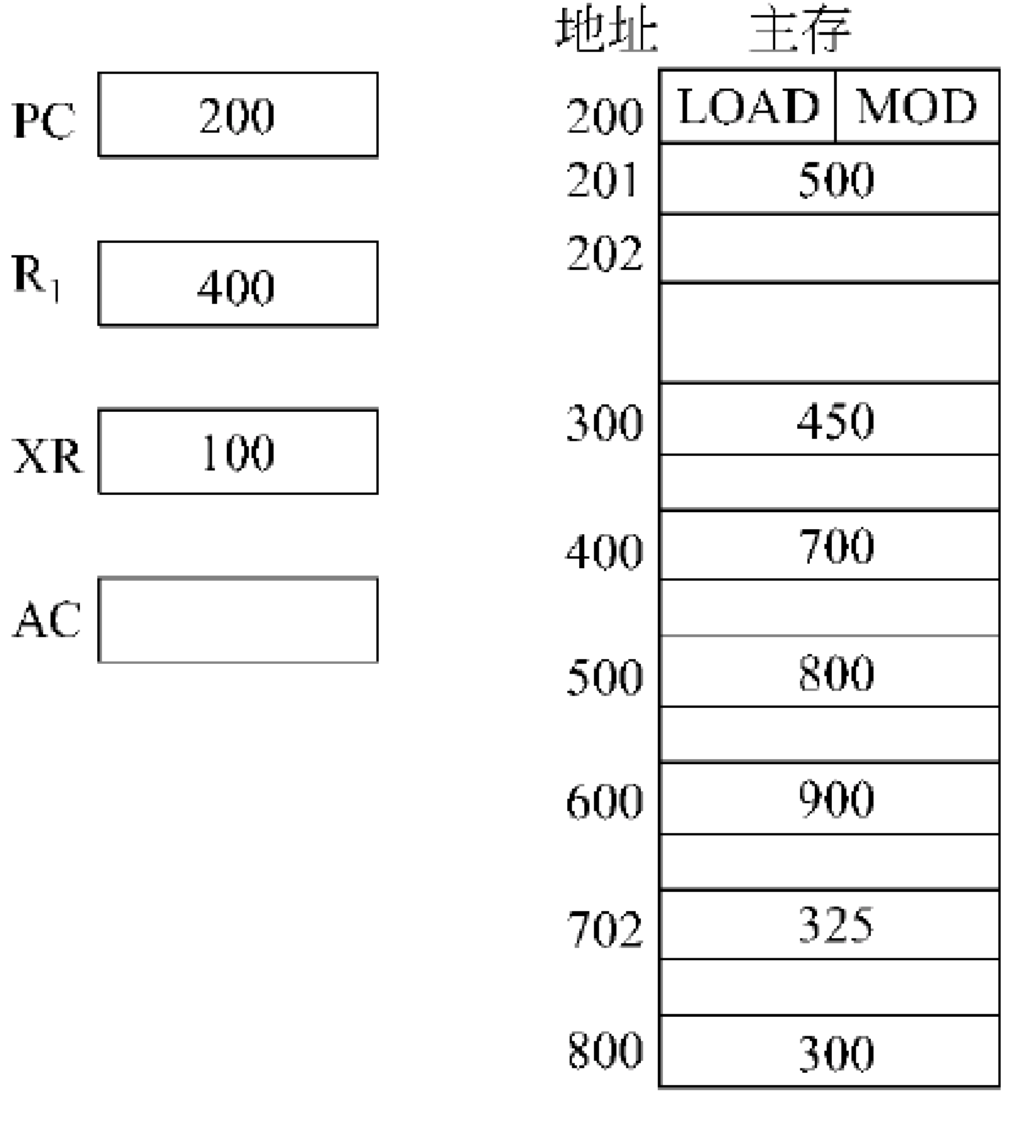


图 3-9　寄存器与主存内容示意图

指令的寻址方式字段可指定任何一种寻址方式。问：在以下几种寻址方式中，装入 AC 的值是多少？

(1) 直接寻址；

(2) 立即寻址；

(3) 间接寻址；

(4) 相对寻址；

(5) 变址寻址；

(6) 寄存器 R_1 寻址；

(7) 寄存器 R_1 间接寻址。

解：(1) 在直接寻址方式下，有效地址是指令中的地址码部分 500，装入 AC 的操作数是 800。

(2) 在立即寻址方式下，指令的地址码部分是操作数而不是地址，所以将 500 装入 AC。

(3) 在间接寻址方式下，操作数的有效地址存储在地址为 500 的单元中，由此得到有效地址为 800，操作数是 300。

(4) 在相对寻址方式下，有效地址 EA＝(PC)＋A＝202＋500＝702，所以操作数是 325。这是因为指令是双字长，在该指令的执行阶段，PC 的内容已经＋2，更新为下一条指令的地址 202。

(5) 在变址寻址方式下，有效地址 EA＝(XR)＋A＝100＋500＝600，所以操作数是 900。

(6) 在寄存器寻址方式下，R_1 的内容 400 装入 AC。

(7) 在寄存器间接寻址方式下，有效地址是 R_1 的内容 400，装入 AC 的操作数是 700。

【例 3.11】　假定指令格式如下：

15　12	11	10	9	8	7　0
OP	I_1	I_2	Z/C	D/I	A

其中：

Bit_{11}＝1，变址寄存器 I_1 寻址；

Bit_{10}＝1，变址寄存器 I_2 寻址；

Bit_9＝1，当前页寻址；

Bit_8(直接/间接寻址)，D/I＝0 表示直接寻址；D/I＝1 表示间接寻址。

若主存容量为 2^{16} 个存储单元，分为 2^8 个页面，每个页面有 2^8 个字。

设有关寄存器的内容为：

$$(I_1)=35A7H\quad (I_2)=1B28H\quad (PC)=46C9H$$

试计算下列指令的有效地址：

(1) D4C1H；

(2) 780BH；

(3) E253H；

(4) C009H。

解:首先将十六进制表示的指令写成二进制,并根据有关的标志位确定不同的寻址方式,最后计算出有效地址。

(1) D4C1H=1101010011000001B

其 Bit_{10}=1,变址寄存器 I_2 寻址,EA=(I_2)+A=1B28H+C1H=1BE9H。

(2) 780BH=0111100000001011B

其 Bit_{11}=1,变址寄存器 I_1 寻址,EA=(I_1)+A=35A7H+0BH=35B2H。

(3) E253H=1110001001010011B

其 Bit_9=1,当前页寻址,因为主存被分为 2^8 个页面,每个页面有 2^8 个字,故页面地址8位,来自于PC的高8位,页内地址8位,来自于指令中的形式地址A。EA=$(PC)_H$//A=46//53=4653H。

(4) C009H=1100000000001001B

有关的标志位均为0,直接寻址 EA=A=0009H。

【例3.12】 某机的指令格式如下:

15　　10	9　　8	7　　　　0
OP	X	A

其中,X为寻址特征位,且X=0时不变址;X=1时用变址寄存器 X_1 进行变址;X=2时用变址寄存器 X_2 进行变址;X=3时相对寻址。设(PC)=1234H,(X_1)=0037H,(X_2)=1122H,请确定下列指令的有效地址(均用十六进制表示)。

(1) 4420H;

(2) 2244H;

(3) 1322H;

(4) 352BH。

解:(1) 指令4420H写成二进制为0100 0100 0010 0000

X=00,不变址,即直接寻址,EA=A=0020H。

(2) 指令2244H写成二进制为0010 0010 0100 0100

X=10,用变址寄存器 X_2 进行变址,EA=(X_2)+A=1122H+44H=1166H。

(3) 指令1322H写成二进制为0001 0011 0010 0010

X=11,相对寻址,EA=(PC)+A=1234H+22H=1256H(未考虑PC值的更新)。

(4) 指令352BH写成二进制为0011 0101 0010 1011

X=01,用变址寄存器 X_1 进行变址,EA=(X_1)+A=0037H+2BH=0062H。

【例3.13】 设相对寻址的转移指令占4个字节,其中,第1、2字节是操作码,第3、4字节是相对位移量(用补码表示)。

(1) 设当前PC的内容为2003H,要求转移到200AH的地址,则该转移指令第3、4字节的内容应为多少?

(2) 设当前PC的内容为2008H,要求转移到2001H的地址,则该转移指令第3、4字节的内容应为多少?

解:由于指令占4个字节,取指令之后(PC)+4。

(1) 第3、4字节的内容为：200AH－(2003H＋4)＝3(补码表示为0003H)。

(2) 第3、4字节的内容为：2001H－(2008H＋4)＝－11(补码表示为FFF5H)。

【例3.14】 以主存地址7EA8H为首地址存放了一条两字节指令，其第一字节为操作码OP，是转移指令；第二字节为相对寻址的位移量 *D*，它是一个8位补码(可正可负)。问：

(1) 位移量 *D* 的表示范围从多少到多少？

(2) 该指令的转移空间可以从哪里到哪里？

解：(1) 位移量 *D* 是8位补码，表示范围－128～127，用十六进制表示为80H～7FH。

(2) 该指令的转移空间是相对于取出该指令之后的PC值＝7EA8H＋2＝7EAAH计算的，7EAAH－80H＝7E2AH，7EAAH＋7FH＝7F29H，所以转移空间为7E2AH～7F29H。

【例3.15】 某一个自底向上生成的存储器堆栈，栈指针始终指向栈顶的满单元。若栈底地址为3000H，栈中已压入两个数据 *a* 和 *b*，SP为堆栈指针。

(1) 试画出此时堆栈的示意图。

(2) 若现在将数据 *c* 和 *d* 按顺序压入堆栈，试写出这两个数据进栈的操作步骤，并画出数据进栈之后堆栈的示意图。

(3) 写出数据 *d* 出栈的操作步骤。

注：设数据交换通过累加器AC进行。

解：(1) 数据 *a* 和 *b* 进栈之后堆栈的示意图如图3-10(a)所示。

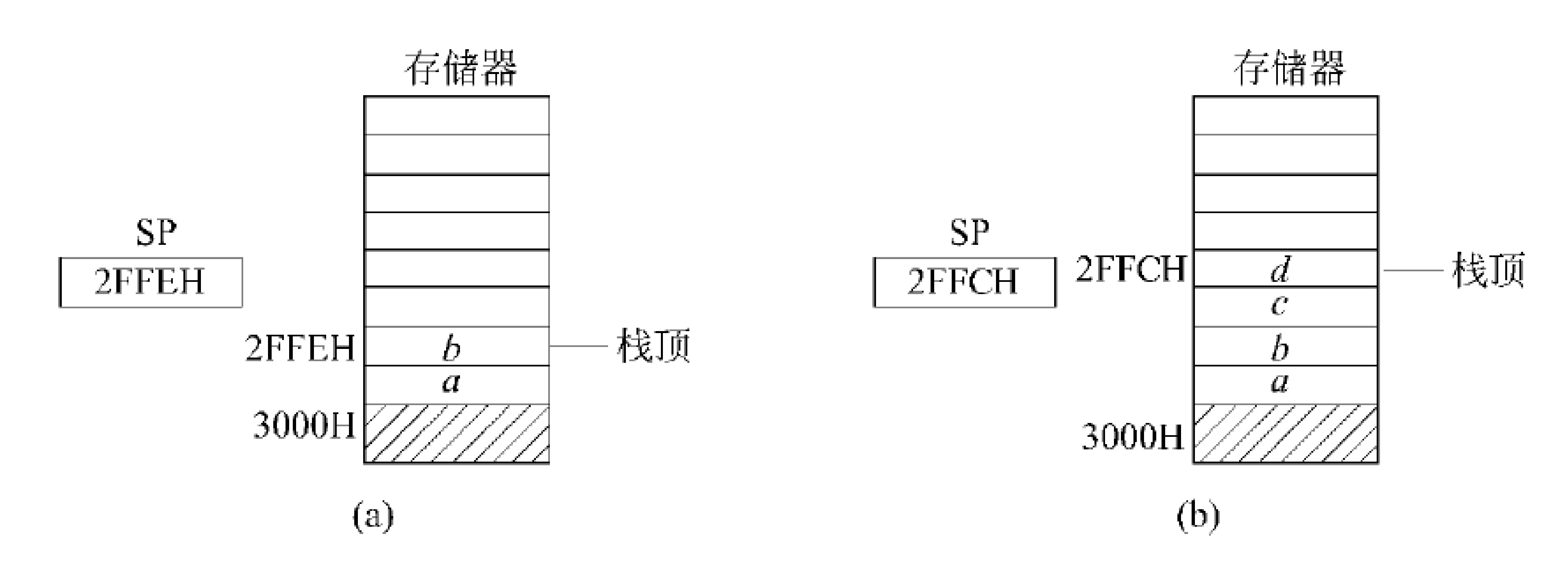


图3-10 堆栈的示意图

(2) 将数据 *c* 压入堆栈的操作步骤为：

c→AC	数据 c 放入累加器 AC
(SP)－1→SP	栈指针－1
(AC)→(SP)	AC的内容压入栈顶单元

将数据 *d* 压入堆栈的操作步骤为：

d→AC	数据 d 放入累加器 AC
(SP)－1→SP	栈指针－1
(AC)→(SP)	AC的内容压入栈顶单元

这两个数据进栈后堆栈的示意图如图3-10(b)所示。

(3) 数据 d 出栈的操作步骤为:

((SP))→AC　　　　将栈顶单元内容弹出送入 AC 中
(SP)+1→SP　　　　栈指针+1

此时数据 d 弹出到累加器 AC 中,再将 AC 的内容写入某个主存单元。

【例 3.16】 存储器堆栈的栈顶内容是 1000H,堆栈自底向上生成,栈指针寄存器 SP 的内容是 100H,一条双字长的子程序调用指令位于存储器地址为 2000H、2001H 处,指令第二个字是地址字段,内容为 3000H。问以下情况下 PC、SP 和栈顶的内容是什么?

(1) 子程序调用指令被读取之前;

(2) 子程序调用指令被执行之后;

(3) 从子程序返回之后。

解:(1) PC 的内容为子程序调用指令的地址,SP 和栈顶的内容在题干中已给出,即有:(PC)=2000H,(SP)=100H,栈顶内容=1000H。

(2) 子程序调用指令被执行之后,PC 内容为子程序入口的指令地址;返回地址进入栈顶,栈指针减 1;由于子程序调用指令为双字长,所以返回地址为子程序调用指令的地址加 2,即 2000H+2=2002H。这样就有:(PC)=3000H,(SP)=FFH,栈顶内容=2002H。

(3) 从子程序返回之后,将返回地址从堆栈中弹出到 PC,这时 SP 加 1,栈顶内容恢复到子程序调用指令被执行之前的值。这样就有:(PC)=2002H,(SP)=100H,栈顶内容=1000H。

【例 3.17】 无条件转移指令和条件转移指令有何不同?转移指令和转子指令又有何不同?

解:无条件转移又称必转,它在执行时将改变程序的常规执行顺序,不受任何条件的约束,直接把程序转向该指令指出的新的位置执行。而条件转移必须受到条件的约束,若条件满足时才执行转移,否则程序仍顺序执行。

转移指令和转子指令都可以改变程序的执行顺序,但转移指令使程序转移到新的地址后继续执行指令,不存在返回的问题,所以没有返回地址;而转子指令要考虑返回问题,所以必须以某种方式保存返回地址。转移指令用于实现同一程序内的转移;而转子指令实现的是程序段与程序段之间的转移。

***【例 3.18】** 某机器字长 16 位,主存按字节编址,转移指令采用相对寻址,由两个字节组成,第一字节为操作码字段,第二字节为相对位移量字段。假定取指令时,每取一个字节 PC 自动加 1。若某转移指令所在主存地址为 2000H,相对位移量字段的内容为 06H,则该转移指令成功转移后的目标地址是________。

A. 2006H　　B. 2007H　　C. 2008H　　D. 2009H

解:C。

分析:主存按字节编址,取指令时,每取一个字节 PC 自动加 1。由于转移指令由两个字节组成,取出这条转移指令之后的 PC 值等于 2002H,所以转移指令成功转移后的目标地址 PC=2000H+2+06H=2008H。

此题容易误选 A 或 B。原因是没有考虑 PC 值的自动更新,或虽然考虑了 PC 要自动更新,但没有注意到这条转移指令是一条 2 字节的指令,PC 值仅仅+1 而不是+2。

*【例 3.19】 偏移寻址通过将某个寄存器内容与一个形式地址相加而生成有效地址。下列寻址方式中,不属于偏移寻址方式的是________。

A. 间接寻址　　B. 基址寻址　　C. 相对寻址　　D. 变址寻址

解: A。

分析: 在 4 种不同的寻址方式中,间接寻址按指令的形式地址从主存中取出操作数的有效地址,然后再按此有效地址从主存中读出操作数。其余 3 种寻址方式可以统称为偏移寻址。

*【例 3.20】 某机器有一个标志寄存器,其中有进位/借位标志(CF)、零标志(ZF)、符号标志(SF)和溢出标志(OF),条件转移指令 bgt(无符号整数比较大于时转移)的转移条件是________。

A. $CF+OF=1$　　B. $\overline{SF}+ZF=1$　　C. $\overline{CF+ZF}=1$　　D. $\overline{CF+SF}=1$

解: C。

分析: 判断无符号整数 $A>B$ 成立,满足的条件是结果不等于 0,即零标志 ZF=0,且不发生进位,即进位/借位标志 CF=0。所以正确选项为 C。其余选项中用到了符号标志 SF 和溢出标志 OF,显然可以排除掉。

*【例 3.21】 假设变址寄存器 R 的内容为 1000H,指令中的形式地址为 2000H;地址 1000H 中的内容为 2000H,地址 2000H 中的内容为 3000H,地址 3000H 的内容为 4000H,则变址寻址方式下访问到的操作数是________。

A. 1000H　　B. 2000H　　C. 3000H　　D. 4000H

解: D。

分析: 变址寻址方式下有效地址 EA=(变址寄存器)+A=(R)+A=1000H+2000H=3000H,操作数 S=(3000H)=4000H。

*【例 3.22】 某计算机有 16 个通用寄存器,采用 32 位定长指令字,操作码字段(含寻址方式位)为 8 位,Store 指令的源操作数和目的操作数分别采用寄存器直接寻址和基址寻址方式。若基址寄存器可使用任一通用寄存器,且偏移量用补码表示,则 Store 指令中偏移量的取值范围是________。

A. −32768~+32767　　B. −32767~+32768

C. −65536~+65535　　D. −65535~+65536

解: A。

分析: 指令字长 32 位,其中操作码字段占 8 位,源/目的寄存器编号各占 4 位,余下 16 位为偏移量(用补码表示),所以偏移量的取值范围为−32768~+32767。

*【例 3.23】 某计算机字长 16 位,主存地址空间大小为 128KB,按字编址,采用单字长指令格式,指令各字段定义如下:

15　12	11　　6		5　　0	
OP	Ms	Rs	Md	Rd

转移指令采用相对寻址方式,相对偏移量用补码表示,寻址方式定义见表 3-8。

表 3-8 例 3.23 题寻址方式定义

Ms/Md	寻址方式	助记符	含义
000B	寄存器直接	R_n	操作数=(R_n)
001B	寄存器间接	(R_n)	操作数=$((R_n))$
010B	寄存器间接、自增	$(R_n)+$	操作数=$((R_n))$,$(R_n)+1\to R_n$
011B	相对	$D(R_n)$	转移目标地址=(PC)+(R_n)

注:(*X*)表示存储器地址 *X* 或寄存器 *X* 的内容。

请回答下列问题:

(1) 该指令系统最多可有多少条指令?该计算机最多有多少个通用寄存器?存储器地址寄存器(MAR)和存储器数据寄存器(MDR)至少各需要多少位?

(2) 转移指令的目标地址范围是多少?

(3) 若操作码 0010B 表示加法操作(助记符为 add),寄存器 R4 和 R5 的编号分别为 100B 和 101B,R4 的内容为 1234H,R5 的内容为 5678H,地址 1234H 的内容为 5678H,地址 5678H 中的内容为 1234H,则汇编语句"add (R4),(R5)+"(逗号前为源操作数,逗号后为目的操作数)对应的机器码是什么(用十六进制表示)?该指令执行后,哪些寄存器和存储单元的内容会改变?改变后的内容是什么?

解:根据指令的格式分析,指令操作码字段占 4 位,源操作数和目的操作数的地址码字段各占 6 位,其中寻址方式占 3 位,寄存器编号占 3 位。给出的寻址方式有寄存器直接寻址、寄存器间接寻址等 4 种。因为主存按字编址,计算机字长为 16 位,主存容量 128KB=64KW。

(1) 指令系统最多可有 $2^4=16$ 条不同的指令,计算机最多可以有 $2^3=8$ 个通用寄存器,主存有 $128\text{KB}\div 2\text{B}=2^{16}$ 个存储单元,故 MDR 和 MAR 至少各需 16 位。

(2) 相对寻址的位移量在通用寄存器 R_n 中,由于 R_n 为 16 位,所以转移指令的目标地址范围是(PC)−32768~(PC)+32767。

(3) 目的操作数采用寄存器间接和自增方式,按照指令格式,汇编语句 add (R4),(R5)+ 对应的机器码为 0010001100010101B=2315H,该指令执行后,寄存器 R5 和地址为 5678H 的存储单元的内容会改变,改变后的内容分别为:

$$(5678H) = ((R4)) + ((R5)) = 5678H + 1234H = 68ACH$$

$$(R5) = 5678H + 1 = 5679H$$

*【例 3.24】 某计算机采用 16 位定长指令字格式,其 CPU 中有一个标志寄存器,其中包括进位/借位标志(CF)、零标志(ZF)和符号标志(NF)。假定为该机设计了条件转移指令,其格式如下:

15 … 11	10	9	8	7 … 0
0 0 0 0 0	C	Z	N	OFFSET

其中,00000 为操作码 OP;C、Z 和 N 分别为 CF、ZF 和 NF 的对应检测位,某检测位为 1 时表示需检测对应标志,需检测的标志位中只要有一个为 1 就转移,否则不转移,例如,若 C=1,Z=0,N=1,则需检测 CF 和 NF 的值,当 CF=1 或 NF=1 时发生转移;OFFSET 是相对

偏移量,用补码表示。转移执行时,转移目标地址为(PC)+2+2×OFFSET;顺序执行时,下条指令地址为(PC)+2。请回答下列问题。

(1) 该计算机存储器按字节编址还是按字编址?该条件转移指令向后(反向)最多可跳转多少条指令?

(2) 某条件转移指令的地址为 200CH,指令内容如下所示,若该指令执行时 CF=0,ZF=0,NF=1,则该指令执行后 PC 的值是多少?若该指令执行时 CF=1,ZF=0,NF=0,则该指令执行后 PC 的值又是多少?请给出计算过程。

15～11	10	9	8	7～0
0 0 0 0 0	0	1	1	1 1 1 0 0 0 1 1

(3) 实现"无符号数比较小于等于时转移"功能的指令中,C、Z 和 N 应各是什么?

(4) 以下是该指令对应的数据通路示意图(图 3-11),要求给出图中部件①～③的名称或功能说明。

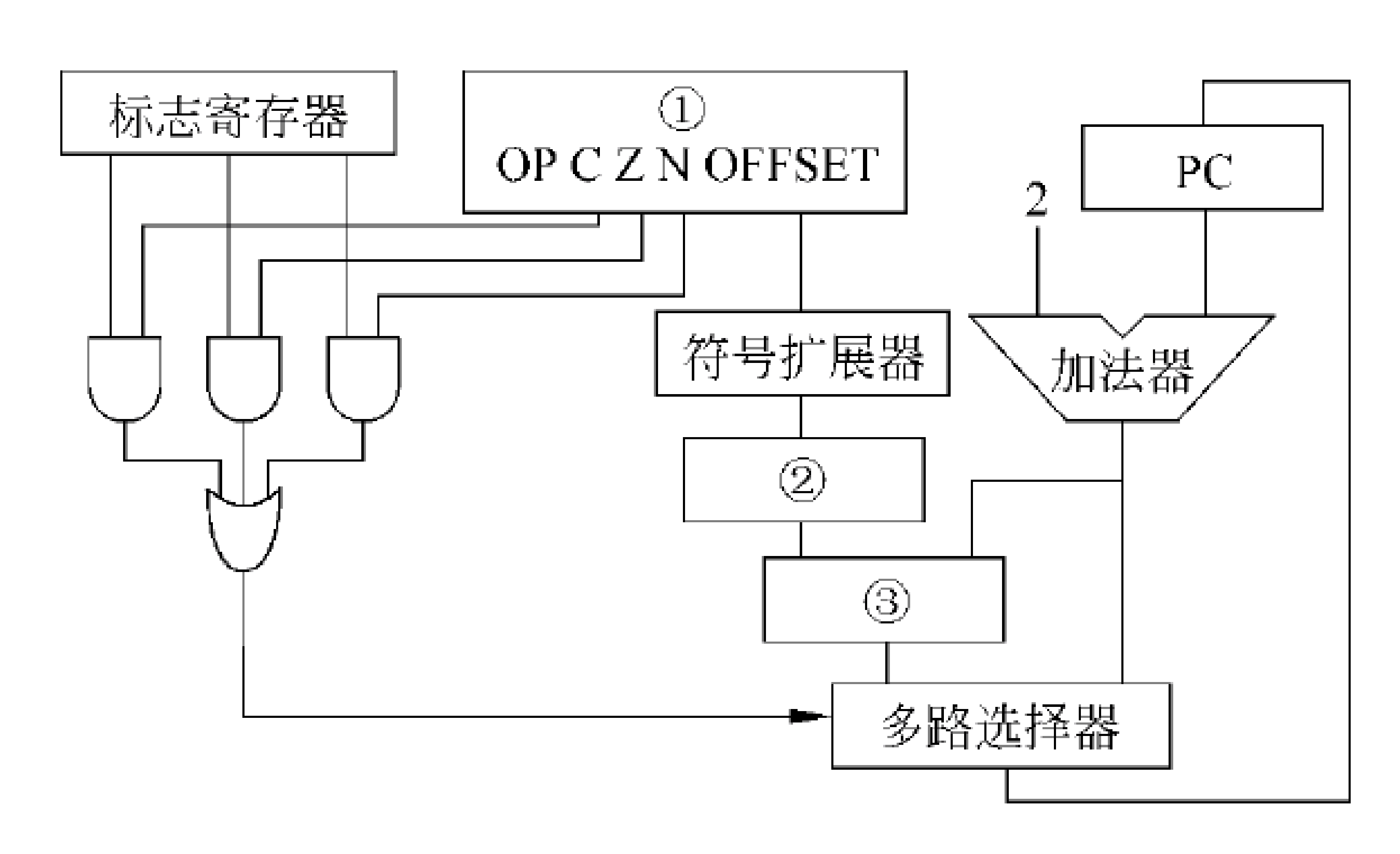


图 3-11　例 3.24 数据通路示意图

解:计算机采用 16 位定长指令字格式,条件转移指令的高 5 位为操作码,中间 3 位为标志寄存器相关标志位的对应检测位,最后 8 位为相对偏移量,用补码表示。条件满足则转移执行,下一条指令地址(转移目标地址)为(PC)+2+2×OFFSET;条件不满足则顺序执行,下一条指令地址为(PC)+2。

(1) 因为指令长度为 16 位,且下一条指令地址为(PC)+2,故编址单位为字节编址而非字编址。由于相对偏移量 OFFSET 为 8 位补码表示,范围为−128～127。向后转移即向反向转移,根据转移地址计算公式,可知(PC)+2+2×OFFSET=(PC)+2+2×(−128)=(PC)−254。由于每条指令占 2 个字节,所以相对于当前条件转移指令,向后(反向)最多可以跳转 127 条指令。

(2) 某条件转移指令的地址为 200CH,指令中 3 个检测位分别为:C=0,Z=1,N=1,所以应根据标志寄存器的标志位 ZF 和 NF 的值决定是否转移。当 CF=0,ZF=0,NF=1 时,转移条件满足,需转移。已知指令中的 8 位偏移量为 11100011=E3H,符号扩展后为 FFE3H,左移一位(乘以 2)后为 FFC6H,所以 PC 的值(即转移目标地址)=(PC)+2+2×OFFSET=200CH+2+FFC6H=1FD4H。当 CF=1,ZF=0,NF=0 时,转移条件不满足,顺序执行。PC 的值为 200CH+2=200EH。

(3) 实现"无符号数比较小于等于时转移"功能,转移条件应满足 CF=1 或 ZF=1,所以

在其指令中,对应检测位C、Z和N应分别设置为C=Z=1,N=0。

(4) 根据图3-11可知,部件①为指令寄存器,其作用为存放从主存中取出的当前指令,部件②为移位寄存器,其作用为左移一位,完成偏移量乘以2的运算(2×OFFSET),部件③为加法器,其作用是地址相加,形成最终的转移地址((PC)+2+2×OFFSET)。

分析:本题是一道涉及多个知识点的综合题,既涉及条件转移指令的转移条件,又涉及指令所对应的数据通路。

注意:必须分清楚标志寄存器中的标志位CF、ZF、NF与指令格式中的检测位C、Z、F的区别,切不可将两者混为一谈。只有指令中对应的检测位为1,且相应的标志位也为1,才能满足转移的条件,从数据通路示意图中是很容易看出这个结论的。

*【例3.25】 某程序中有如下循环代码段P:for(i=0;i<N;i++)sum+=A[i];,假设编译时变量sum和i分别分配在寄存器R1和R2中,常量N在寄存器R6中,数组A的首地址在寄存器R3中,程序段P起始地址为0804 8100H,对应的汇编代码和机器代码如表3-9所示。

表3-9 程序段P

编号	地 址	机器代码	汇编代码	注 释
1	08048100H	00022080H	Loop: sll R4,R2,2	(R2)<<2→R4
2	08048104H	00832020H	add R4,R4,R3	(R4)+(R3)→R4
3	08048108H	8C850000H	load R5,0(R4)	((R4)+0)→R5
4	0804810CH	00250820H	add R1,R1,R5	(R1)+(R5)→R1
5	08048110H	20420001H	addi R2,R2,1	(R2)+1→R2
6	08048114H	1446FFFAH	bne R2,R6,loop	if(R2)!=(R6) goto loop

执行上述代码的计算机M采用32位定长指令字,其中分支指令bne采用如下格式。

31 26	25 21	20 16	15 0
OP	Rs	Rd	OFFSET

OP为操作码;Rs和Rd为寄存器编号;OFFSET为偏移量,用补码表示。请回答下列问题,并说明理由。

(1) M的存储器编址单位是什么?

(2) 已知sll指令实现左移功能,数组A中每个元素占多少位?

(3) 表3-9中bne指令的OFFSET字段的值是多少?已知bne指令采用相对寻址方式,当前PC内容为bne指令地址,通过分析表3-9中指令地址和bne指令内容,推断出bne指令的转移目标地址计算公式。

(4) 若M采用如下"按序发射、按序完成"的5级指令流水线:IF(取指)、ID(译码及取数)、EXE(执行)、MEM(访存)、WB(写回寄存器),且硬件不采取任何转发措施,分支指令的执行均引起3个时钟周期的阻塞,则P中哪些指令的执行会由于数据相关而发生流水线阻塞?哪条指令的执行会发生控制冒险?为什么指令1的执行不会因为与指令5的数据相关而发生阻塞?

解：首先需要分析清楚程序段 P 所对应的这 6 条指令的作用。指令 1(sll R4,R2,2)是一条左移 2 位指令，它完成将 i 值乘以 4；指令 2(add R4,R4,R3)是一条加法指令，它的作用是将 $4i$ 加上数组 A 的首地址，以产生数组 A 每个元素的地址；指令 3(load R5,0(R4))是一条取数指令，通过寄存器间接寻址，把数组 A 的某个元素从主存中取出来；指令 4(add R1,R1,R5)是真正完成求和的加法指令；指令 5(addi R2,R2,1)实现 $i+1$；指令 6(bne R2,R6,loop)是一条条件转移指令，当(R2) ≠ (R6)时，转移至第一条指令，否则从循环中出来。当对每一条指令的作用都搞清楚之后，可以比较容易的回答下列问题：

(1) 存储器编址单位是字节。因为从表 3-9 可以看出，每条指令占 4 个单元，指令长度为 32 位。

(2) 数组 A 中每个元素占 32 位。因为数组 A 每个元素的地址通过下标左移 2 位(即乘 4)再加数组首址得到，故每个数组元素占 4 个字节，即 32 位。

(3) 指令 bne 的转移目标地址计算公式为：(PC)＋4＋OFFSET×4。因为根据指令 bne 的机器代码可以得知 OFFSET＝FFFAH，偏移量用补码表示，十进制真值为－6。又从表 3-9 看到，指令 bne 所在地址(PC)为 08048114H，故转移目标地址为 08048100H，08048100H＝08048114H＋4＋(－6)×4，故转移目标地址计算公式为(PC)＋4＋OFFSET×4。

(4) 由于数据相关而发生阻塞的指令为第 2、3、4、6 条，因为第 2、3、4、6 条指令都与各自的前一条指令发生数据相关。第 6 条指令会发生控制冒险，因为这是一条条件指令。在循环的过程中，当前循环的第 5 条指令与下次循环的第 1 条指令也存在着数据相关，但由于第 6 条指令后面有 3 个时钟周期的阻塞，因而消除了该数据相关。

分析：本题是一道涉及多个知识点的综合题。需要清楚高级语言、汇编语言、机器语言之间的关系，以及机器代码在存储器中存放的情况，同时还需要对指令流水线的概念非常清楚。其中第(1)、(2)小题难度不高，但第(3)、(4)小题难度较大。

3.4 同步测试习题及解答

3.4.1 同步测试习题

一、填空题

1. 零地址运算指令的操作数来自________。

2. 根据操作数所在位置，指出其寻址方式：操作数在寄存器中，称为________寻址方式；操作数地址在寄存器中，称为________寻址方式；操作数在指令中，称为________寻址方式；操作数地址在指令中，称为________寻址方式。操作数的地址，为某一个寄存器中的内容与位移之和，则可以是________、________和________寻址方式。

3. 设字长和指令长度均为 24 位，若指令系统可完成 108 种操作，并且具有直接、间接(一次间址)、变址、基址、相对、立即 6 种寻址方式，则在保证最大范围内直接寻址的前提下，指令字中操作码占________位，寻址特征位占________位，可直接寻址的范围是________，一次间址的范围是________。

二、选择题

1. 执行一条一地址的加法指令共需要________次访问主存(含取指令)。

A. 1　　B. 2　　C. 3　　D. 4

2. 零地址的运算类指令在指令格式中不给出操作数地址，参加的两个操作数来自________。

A. 累加器和寄存器　　B. 累加器和暂存器

C. 堆栈的栈顶和次栈顶单元　　D. 暂存器和堆栈的栈顶单元

3. 在关于一地址运算类指令的叙述中，正确的是________。

A. 仅有一个操作数，其地址由指令的地址码提供

B. 可能有一个操作数，也可能有两个操作数

C. 一定有两个操作数，另一个是隐含的

D. 指令的地址码字段存放的一定是操作码

4. 一个计算机系统采用32位单字长指令，地址码为12位，如果定义了250条二地址指令，那么单地址指令的条数有________。

A. 4K　　B. 8K　　C. 16K　　D. 24K

5. 某计算机存储器按字(16位)编址，每取出一条指令后PC值自动+1，说明其指令长度是________。

A. 1个字节　　B. 2个字节　　C. 3个字节　　D. 4个字节

6. 一条指令有128位，按字节编址，读取这条指令后，PC的值自动加________。

A. 1　　B. 2　　C. 4　　D. 16

7. 在寄存器间接寻址方式中，操作数应该在________中。

A. 寄存器　　B. 堆栈栈顶　　C. 累加器　　D. 主存单元

8. 直接、间接、立即3种寻址方式指令的执行速度，由快至慢的排序是________。

A. 直接、立即、间接　　B. 直接、间接、立即

C. 立即、直接、间接　　D. 立即、间接、直接

9. 为了缩短指令中某个地址码的位数，而指令的执行时间又相对短，则有效的寻址方式是________。

A. 立即寻址　　B. 寄存器寻址　　C. 直接寻址　　D. 寄存器间接寻址

10. 指令操作所需的数据不会来自________。

A. 寄存器　　B. 指令本身　　C. 主存　　D. 控制存储器

11. 在变址寄存器寻址方式中，若变址寄存器的内容是4E3CH，指令中的形式地址是63H，则它对应的有效地址是________。

A. 63H　　B. 4D9FH　　C. 4E3CH　　D. 4E9FH

12. 设变址寄存器为X，形式地址为D，某机具有先间址后变址的寻址方式，则这种寻址方式的有效地址为________。

A. $EA=(X)+D$　　B. $EA=(X)+(D)$

C. $EA=((X)+D)$　　D. $EA=X+D$

13. 采用变址寻址可以扩大寻址范围，且________。

A. 变址寄存器的内容由用户确定，在程序执行过程中不能改变

B. 变址寄存器的内容由操作系统确定，在程序执行过程中不能改变

C. 变址寄存器的内容由用户确定，在程序执行过程中可以改变

D. 变址寄存器的内容由操作系统确定,在程序执行过程中可以改变

14. 变址寻址和基址寻址的有效地址形成方式类似,但________。

A. 变址寄存器的内容在程序执行过程中是不能改变的

B. 基址寄存器的内容在程序执行过程中是可以改变的

C. 在程序执行过程中,变址寄存器的内容不能改变而基址寄存器的内容可变

D. 在程序执行过程中,基址寄存器的内容不能改变而变址寄存器的内容可变

15. 用来支持浮动程序设计的寻址方式是________。

A. 相对寻址　　B. 变址寻址

C. 寄存器间接寻址　　D. 基址寻址

16. 设相对寻址的转移指令占两个字节,第一个字节是操作码,第二个字节是相对位移量(用补码表示)。每当 CPU 从存储器取出第一个字节时,即自动完成(PC)+1→PC。设当前 PC 的内容为 2003H,要求转移到 200AH 地址,则该转移指令第二字节的内容应为________。若 PC 的内容为 2008H,要求转移到 2001H,则该转移指令第二字节的内容应为________。

A. 05H　　B. 06H　　C. 07H　　D. F7H

E. F8H　　F. F9H

17. 在存储器堆栈中,保持不变的是________。

A. 栈顶　　B. 栈指针　　C. 栈底　　D. 栈中的数据

18. 堆栈寻址方式中,设 A 为累加器,SP 为堆栈指示器,M_{SP}为 SP 指示的栈顶单元,如果进栈操作的动作顺序是(A)→M_{SP},(SP)－1→SP,那么出栈操作的动作顺序应为________。

A. (M_{SP})→A,(SP)+1→SP　　B. (SP)+1→SP,(M_{SP})→A

C. (SP)－1→SP,(M_{SP})→A　　D. 以上都不对

19. 要想使 8 位寄存器 A 中的高 4 位变为 1,低 4 位不变,可使用________。

A. A∨0FH→A　　B. A∧0FH→A

C. A∧F0H→A　　D. A∨F0H→A

注:∧表示“与”指令,∨表示“或”指令

20. 程序控制类指令的功能是________。

A. 进行主存和 CPU 之间的数据传送　　B. 进行 CPU 和外设之间的数据传送

C. 改变程序执行的顺序　　D. 控制进栈、出栈操作

21. 下列不属于程序控制指令的是________。

A. 无条件转移指令　　B. 条件转移指令

C. 中断隐指令　　D. 循环控制指令

22. 将子程序返回地址放在________中时,子程序允许嵌套和递归。

A. 寄存器　　B. 堆栈

C. 子程序的结束位置　　D. 子程序的起始位置

23. I/O 编址方式通常可分统一编址和独立编址,________。

A. 统一编址是将 I/O 地址看作是存储器地址的一部分,可用专门的 I/O 指令对设备进行访问

B. 独立编址是指I/O地址和存储器地址是分开的,所以对I/O访问必须有专门的I/O指令

C. 统一编址是指I/O地址和存储器地址是分开的,所以可用访存指令实现CPU对设备的访问

D. 独立编址是将I/O地址看作是存储器地址的一部分,所以对I/O访问必须有专门的I/O指令

三、判断题

1. 数据寻址的最终目的是寻找操作数的有效地址。( )
2. 若操作数在寄存器中,可以采用直接寻址。( )
3. 在一条机器指令中可能出现不止一种寻址方式。( )
4. 寄存器堆栈的栈指针SP指向栈顶。( )
5. 对于自底向上生成的软堆栈,进栈时应先修改栈指针,再将数据压入堆栈。( )
6. 进栈操作是指将内容写入堆栈指针SP。( )
7. 不设置浮点运算指令的计算机,就不能用于科学计算。( )
8. 转子指令是一条零地址指令。( )
9. 返回指令通常是一条零地址指令。( )
10. 转移类指令能改变指令执行顺序,因此执行这类指令时,PC和SP的值都将发生变化。( )

四、简答题

1. 在寄存器—寄存器型、寄存器—存储器型和存储器—存储器型3类指令中,哪类指令的执行时间最长?哪类指令的执行时间最短?为什么?

2. 简述立即寻址方式的特点。

五、综合题

1. 某计算机的指令系统定长为16位,采用扩展操作码,操作数地址需4位。该指令系统已有三地址指令M条,二地址指令N条,没有零地址指令。问:最多还有多少条一地址指令?

2. 某机器指令码长度16位,地址码长度都为6位,包含单地址指令、双地址指令和零地址指令,试问单地址指令最多能有多少条?此时双地址指令和零地址指令各为多少条?

3. 设计算机A有60条指令,指令操作码为6位固定长度编码,从000000到111011。其后继产品B需要增加32条指令,并与A保持兼容。

(1) 试采用操作码扩展技术为计算机B设计指令操作码。

(2) 计算操作码的平均长度。

4. 设计算机指令字长为16位,指令中地址字段的长度为4位,共11条三地址指令,72条二地址指令,64条零地址指令。问最多还能安排多少条一地址指令?

5. 某机字长16位,主存容量为64KB,指令为单字长指令,有50种操作码,采用页面、间接和直接寻址方式。

(1) 指令格式如何安排?

(2) 存储器能划分为多少页面?每页多少单元?

(3) 能否再增加其他寻址方式?

6．某机主存容量为4M×16，且存储字长等于指令字长，若该机指令系统可完成108种操作，操作码位数固定，且具有直接、间接、变址、基址、相对、立即6种寻址方式，试回答：

（1）画出一地址指令格式，并指出各字段的作用；

（2）该指令直接寻址的最大范围；

（3）一次间址和多次间址的寻址范围；

（4）立即数的范围（十进制表示）；

（5）相对寻址的位移量（十进制表示）；

（6）上述6种寻址方式的指令哪一种执行时间最短？哪一种执行时间最长？为什么？哪一种便于程序浮动？哪一种最适合处理数组问题？

7．某16位机器所使用的指令格式和寻址方式如图3-12所示，该机有2个20位基址寄存器，4个16位变址寄存器，16个16位通用寄存器，指令汇编格式中的S(源)，D(目标)都是通用寄存器，M是主存的一个单元，三种指令的操作码分别是MOV(OP)＝0AH，STA(OP)＝1BH，LDA(OP)＝3CH，其中MOV为传送指令，STA为写数指令，LDA为读数指令。

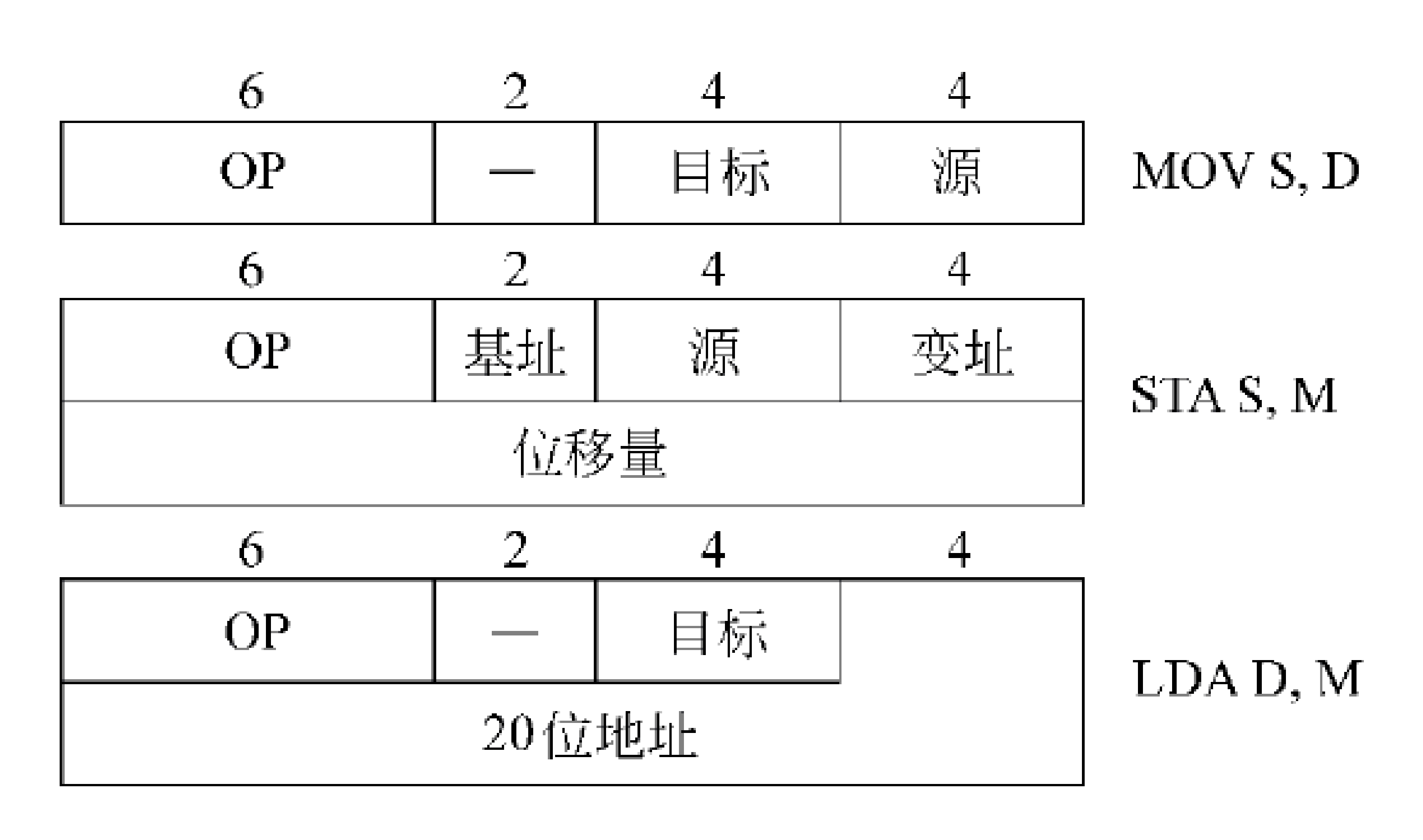


图3-12　综合题习题7的指令格式

要求：（1）分析3种指令的指令格式和寻址方式特点。

（2）CPU完成哪一种操作所花费的时间最短？哪一种操作所花费的时间最长？第二种指令的执行时间有时会等于第三种指令的执行时间吗？

（3）下列情况下每个十六进制指令字分别代表什么操作？其中有编码不正确时，如何改正才能成为合法指令？

①F0F1H，3CD2H　　②2856H　　③6FD6H　　④1C2H

六、设计题

1．CPU的双操作数指令格式如图3-13所示。

OP为4位操作码；Md和Ms分别为3位目的和源操作数寻址方式；Rd和Rs分别为3位目的和源寄存器号。问：

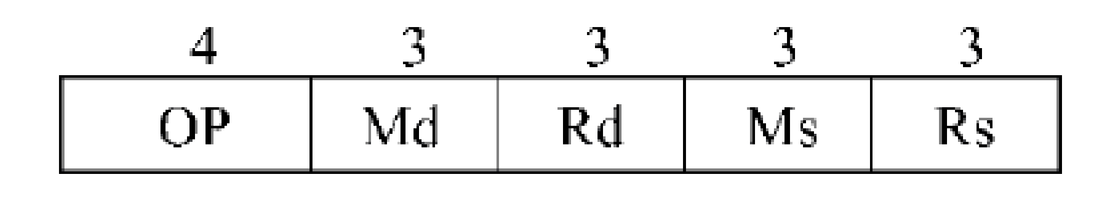


图3-13　CPU的双操作数指令格式

（1）计算机设计16种双操作数指令是否可取？为什么？

（2）CPU内部寄存器增加到16个，在不改变指令长度的条件下，可以用哪两种方式修改指令格式（画出修改后的指令格式），将对指令功能产生什么影响？

(3) 如不降低指令功能,指令长度可变,画出具有16个寄存器的双操作数指令的格式。

2. 某机字长16位,直接寻址空间128字,变址时的位移量是−64～+63,16个通用寄存器都可以作为变址寄存器,设计一套指令系统,满足下列寻址类型的要求。

(1) 直接寻址的二地址指令3条;

(2) 变址寻址的一地址指令6条;

(3) 寄存器寻址的二地址指令8条;

(4) 直接寻址的一地址指令12条;

(5) 零地址指令32条。

3.4.2 同步测试习题解答

一、填空题

1. 堆栈。

2. 寄存器,寄存器间接,立即,直接,变址,基址,相对。

3. 7,3,2^{14},2^{24}。指令系统可完成108种操作,至少需要地址码7位,6种寻址方式至少需要3位寻址特征位,所以指令中的形式地址部分只剩下14位,可直接寻址的范围为2^{14}。而一次间址时,根据形式地址找到的有效地址有24位,所以一次间址的范围是2^{24}。

二、选择题

1. B。执行一条一地址的加法指令,只需要访问两次主存。第一次取指令本身,第二次取第二操作数。第一操作数和运算结果都放在累加寄存器中,所以读取和存入都不需要访问主存。

2. C。零地址的运算类指令即堆栈运算指令,参加的两个操作数来自堆栈的栈顶和次栈顶。

3. B。一地址运算类指令包括单操作数指令(如加1、减1指令)和双操作数指令(如加、减指令)两类。对于单操作数指令只需要一个操作数,对于双操作数指令需要两个操作数,其中一个操作数的地址是显地址,另一个操作数隐含在累加寄存器中。

4. D。二地址指令的操作码字段8位,现定义了250条二地址指令,采用扩展操作码技术,留下6个扩展窗口,每个扩展窗口可以扩展$2^{12}=4K$条一地址指令,故共可扩展$6\times 4K=24K$条一地址指令。

5. B。存储器按字编址,每取一条指令PC+1,说明指令长16位。

6. D。指令长128位(16个字节),每取出一条指令,PC+16。

7. D。寄存器间接寻址,寄存器中存放的是操作数的有效地址,而操作数在主存中。

8. C。这3种寻址方式由快至慢的顺序是:立即寻址、直接寻址、间接寻址。

9. B。寄存器寻址方式最显著的优点就是:①从寄存器中存取数据比从主存中快得多。②由于寄存器的数量较少,其地址码字段比主存单元地址字段短得多。

10. D。指令操作所需的数据可能来自于指令本身,也可能来自于寄存器或主存单元,但不会来自控制存储器。

11. D。$EA=(R_x)+A=4E3CH+63H=4E9FH$

12. B。A选项是变址寻址,C选项为先变址后间址,D选项的表达式不对。

13. C。变址寻址是面向用户的,用于访问字符串、向量和数组等成批数据,变址寄存器

的内容在程序执行过程中可以修改。

14. D。变址寻址中变址寄存器提供修改量(可变的),而指令中提供基准值(固定的);基址寻址中基址寄存器提供基准值(固定的),而指令中提供位移量(可变的)。

15. A。相对寻址方式编写的程序可在主存中任意浮动,它放在主存的任何地方,所执行的效果都是一样的。

16. A,D 。由于转移指令占两个字节,当 PC 的内容为 2003H 时,取出转移指令后 PC 的内容为 2005H,所以有 200AH－2005H＝05H。当 PC 的内容为 2008H 时,取出转移指令后 PC 的内容为 200AH,所以有 2001H－200AH＝－9H,用补码表示为 F7H。

17. C。存储器堆栈的大小可变,栈底固定,栈顶浮动。

18. B。如果进栈时是先压入数据,说明栈指针是指向栈顶的空单元,所以出栈时就要先修改栈指针,然后才能弹出数据。

19. D。利用“或”指令可以使目的操作数的某些位置“1”。

20. C。程序控制类指令用于控制程序的执行顺序,并使程序具有测试、分析与判断的能力。

21. C。中断隐指令没有操作码,它并非真正的指令,更不是程序控制指令。

22. B。只有将返回地址存放在堆栈中,才能不仅允许子程序嵌套而且允许子程序递归。

23. B。统一编址是将 I/O 地址看作是存储器地址的一部分,不需要专门的 I/O 指令。

三、判断题

1. ×。数据寻址的最终目的是寻找操作数。
2. ×。若操作数在寄存器中,采用寄存器寻址。
3. √。
4. ×。寄存器堆栈无须栈指针 SP。
5. √。
6. ×。进栈操作是将内容写入栈顶,其栈顶地址由栈指针 SP 提供。
7. ×。不设置浮点运算指令的计算机,仍可用于科学计算,只是要增加编程量且速度并不快。
8. ×。转子指令中必须给出子程序的首地址,所以一定是一条一地址指令。
9. √。
10. ×。执行这类指令时,SP 的值不会发生变化。

四、简答题

1. 寄存器-寄存器型指令执行速度最快,存储器-存储器型指令执行速度最慢。因为前者操作数在寄存器中,后者操作数在存储器中,而访问一次存储器所需要的时间比访问一次寄存器所需要的时间长。

2. 立即寻址方式的特点是执行速度快,取指令的同时也取出数据,不需要寻址计算和访问主存,但操作数是固定不变的,因此适合于访问常数。

五、综合题

1. 一地址指令最多还有$((2^4-M)\times 2^4-N)\times 2^4=2^{12}-M\times 2^8-N\times 2^4$条。

2. 单地址指令最多能有$(2^4-1)\times 2^6-1=959$条。此时双地址指令只有 1 条,零地址

指令最多可有 $2^6=64$ 条。

3. (1) 6位操作码中保留了从111100到111111共4个扩展窗口,将它们扩展成9位操作码,可扩展32条指令(4×8=32),为保证与计算机A的指令兼容,新增加的32条指令的操作码从111100000到111111111。

(2) 操作码的平均长度=(60×6+32×9)÷(60+32)=7.04。

4. 三地址指令只有4位操作码,现有11条三地址指令,所以还有16-11=5个扩展窗口用于二地址指令。二地址指令有8位操作码,去掉三地址指令用掉的操作码,可规定5×16=80条二地址指令,现有72条二地址指令,所以还有80-72=8个扩展窗口用于一地址指令。一地址指令有12位操作码,可规定8×16=128条一地址指令。但要求有64条零地址指令,所以需要由一地址指令提供给零地址指令64÷16=4个扩展窗口,因此最多还能安排128-4=124条一地址指令。

5. (1) 由于机器字长16位,指令为单字长指令(16位)。现在有50种不同的操作码,需要操作码字段6位;寻址方式有3种,寻址方式字段2位;剩下的8位为地址码字段。

(2) 若采用页面寻址,需将存储器划分成若干页面。主存容量共64KB,需要地址16位。已知指令中的地址字段(页内地址)为8位,则页面地址也有8位(16-8=8),故存储器能划分256个页面,每一页面有256个单元。

(3) 在(1)题中确定的指令格式情况下,还可以再增加一种寻址方式。因为寻址方式字段有2位,允许出现4种不同的寻址方式。

6. (1) 一地址指令格式如图3-14所示,各字段的作用为如下。

OP:操作码字段,指定操作类型;

MOD:寻址方式字段,指定寻址方式;

A:地址码字段,指定操作数地址或操作数。

15 ~ 9	8 ~ 6	5 ~ 0
OP	MOD	A

图3-14 一地址指令格式

(2) 直接寻址的最大范围为 2^6。因为操作码字段占7位,寻址方式字段占3位,所以地址码字段长6位。直接寻址的范围为 $2^{16-7-3}=2^6=64$ 个单元。

(3) 间接寻址有一次间址和多次间址之分,一次间址的寻址范围为 $2^{16}=65536$ 个单元;多次间址的寻址范围为 $2^{15}=32768$ 个单元。

(4) 十进制表示立即数的范围为-32~31(补码时)或-31~31(原码时)。

(5) 十进制表示相对寻址的位移量为-32~31(补码时)或-31~31(原码时)。

(6) 在上述几种寻址方式中,立即寻址指令执行时间最短,间接寻址(多次间址)指令执行时间最长。相对寻址方式便于实现程序浮动,变址寻址方式最适合处理数组问题。

7. (1) 第一种指令是单字长二地址指令,R-R型,寄存器寻址;第二种指令是双字长二地址指令,R-M型,其中R由源寄存器决定,M采用基址寻址或变址寻址;第三种指令也是双字二地址指令,R-M型,其中R由目标寄存器决定,M由20位地址(直接寻址)决定。

(2) CPU完成第一种指令所花的时间最短,因为是R-R型指令,除掉取指令之外不需要访问存储器。第二种指令所花费的时间最长,因为是R-M型指令,需要访问存储器,同时还要进行寻址方式的变换运算(基址或变址)。第二种指令的执行时间不会等于第三种指令,因为第三种指令虽也访问存储器,但节省了求有效地址运算的时间开销。

(3) 指令操作码采用定长编码(6位),根据已知条件:MOV(OP)=001010,STA(OP)

=011011，LDA(OP)=111100。

指令的十六进制格式转换成二进制代码且比较后可知：

① 将 F0F1H 和 3CD2H 的前 6 位转换成二进制代码可以发现这是一条 LDA 指令，编码正确，其含义是把主存 13CD2H 地址单元的内容取至第 15 号通用寄存器中。

② 将 2856H 的前 6 位转换成二进制代码可以发现这是一条 MOV 指令，编码正确，含义是把第 6 号通用寄存器(源)的内容传送至第 5 号通用寄存器(目标)中。

③ 由于 6FD6H 是单字长指令，一定是 MOV 指令，但编码错误，可将其改正为 28D6H。

④ 1C2H 也是单字长指令，但编码错误，可改正为 28C2H，代表 MOV 指令。

六、设计题

1. (1) 如果计算机中仅有双操作数指令，最多允许设计 16 条，因为操作码字段有 4 位。但通常考虑到指令系统中还有单操作数指令和无操作数指令，所以应当留出一些扩展窗口供它们扩展操作码使用。

(2) 当 CPU 内部寄存器增加到 16 个，在不改变指令长度的条件下，可以用以下两种方式修改指令格式。

方式 1：减少操作码字段长度来增加 Rd 和 Rs 的长度，这种方式将减少双操作数指令的条数，如图 3-15(a)所示。

方式 2：减少 Md 和 Ms 的长度来增加 Rd 和 Rs 的长度，这种方式将减少寻址方式。如图 3-15(b)所示。

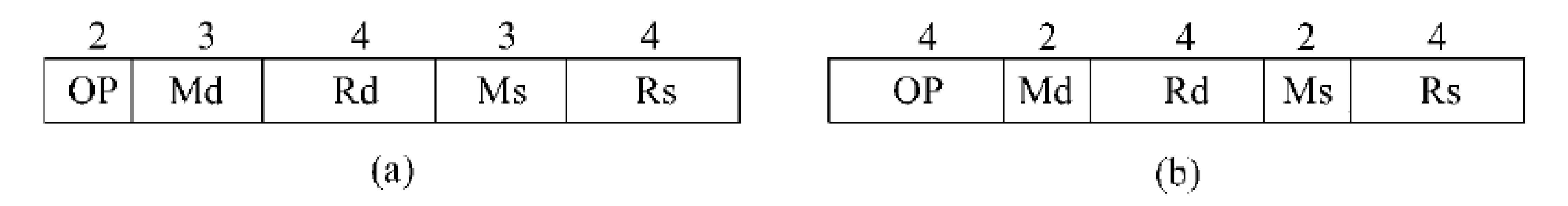


图 3-15　不改变指令长度条件下的修改方案

(3) 如不降低指令功能，指令长度可变，则指令长度增加为 18 位。图 3-16 画出具有 16 个寄存器的双操作数指令的格式。

2. 5 种类型的指令格式如图 3-17 所示。

4	3	4	3	4
OP	Md	Rd	Ms	Rs

图 3-16　改变指令长度条件下的修改方案

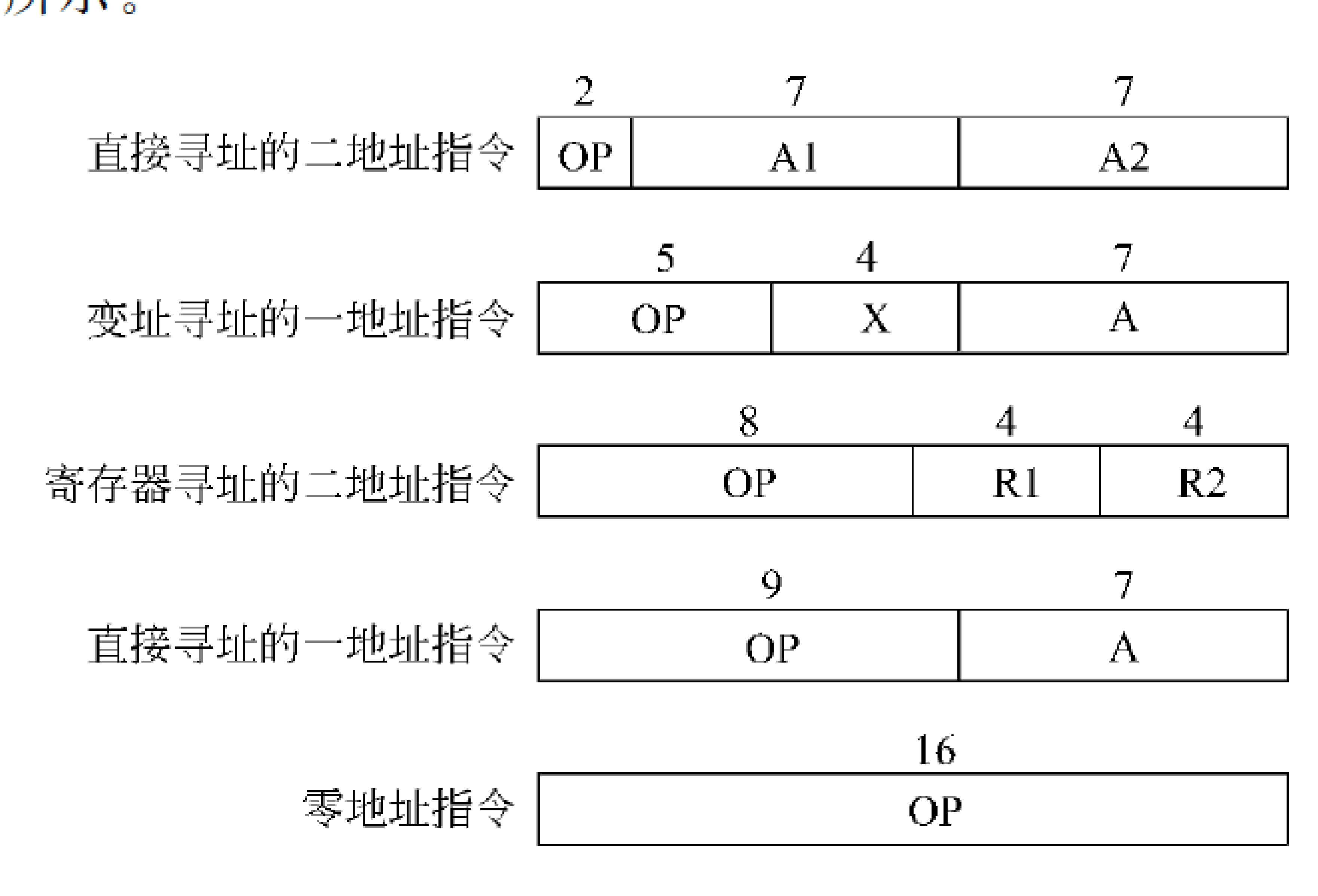


图 3-17　5 种类型的指令格式

(1) 直接寻址的二地址指令3条,其操作码编码为:

00

01

10

(2) 变址寻址的一地址指令6条,其操作码编码为:

11000

⋮

11101

(3) 寄存器寻址的二地址指令8条,其操作码编码为:

11110000

⋮

11110111

(4) 直接寻址的一地址指令12条,其操作码编码为:

111110000

⋮

111111011

(5) 零地址指令32条,其操作码编码为:

1111111000000000

⋮

1111111000011111

第 4 章 数值的机器运算

4.1 基本内容摘要

- 基本算术运算的实现
  - 加法器
  - 进位的产生和传递
  - 并行加法器的快速进位

    并行进位方式；

    分组并行进位方式。
- 定点加减运算
  - 原码加减运算
  - 补码加减运算

    补码加法；

    补码减法。
  - 补码的溢出判断与检测方法

    溢出的产生；

    溢出检测方法。
  - 补码定点加减运算的实现
- 带符号数的移位和舍入操作
  - 带符号数的移位操作
  - 带符号数的舍入操作
- 定点乘法运算
  - 原码一位乘法
  - 补码一位乘法
  - 补码两位乘法
- 定点除法运算
  - 原码除法运算
  - 补码除法运算
- 规格化浮点运算
  - 浮点加减运算

- ◆ 浮点乘除运算
- ◆ 浮点运算器的实现
- ● 十进制整数的加法运算
  - ◆ 一位十进制加法运算
  - ◆ 十进制加法器
- ● 逻辑运算与实现
- ● 运算器的基本组成与实例
  - ◆ 运算器结构
  - ◆ ALU 举例
  - ◆ 浮点运算器举例

4.2 重点难点梳理

1. 串行加法器与并行加法器

加法器有串行和并行之分。在串行加法器中,只有一个全加器,使用移位寄存器从低位到高位串行地提供操作数并送入全加器进行运算,对于 n 位字长的加法,分 n 步进行相加。并行加法器则由多个全加器组成,n 位字长的加法器,由 n 个全加器组成,n 位数据同时相加。

串行加法器具有器件少、成本低的优点,但运算速度太慢,所以除去某些低速的专用运算器外很少采用。

并行加法器可以同时对数据的各位相加,但存在着一个加法的最长运算时间问题。这是因为虽然操作数的各位是同时提供的,但低位运算所产生的进位会影响高位的运算结果。例如:

$$\begin{array}{r} 1\,1\cdots1\,1 \\ +\quad 0\,0\cdots0\,1 \\ \hline 1\,0\,0\cdots0\,0 \end{array}$$

最低位产生的进位将逐位影响至最高位,因此,并行加法器的最长运算时间主要是由进位信号的传递时间决定的,而每个全加器本身的求和延迟只是次要因素。很明显,提高并行加法器速度的关键是尽量加快进位产生和传递的速度。

2. 并行加法器的进位方式

并行加法器中的每一个全加器都有一个从低位送来的进位和一个传送给较高位的进位,每一位的进位表达式为:

$$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$$

其中,A_iB_i 称为进位产生函数(本次进位产生),用 G_i 表示;$A_i \oplus B_i$ 称为进位传递函数(低位进位传递),用 P_i 表示。

1) 串行进位方式

串行进位方式的每一级进位直接依赖于前一级的进位,即进位信号是逐级形成的。

$$C_1 = G_1 + P_1C_0$$
$$C_2 = G_2 + P_2C_1$$

$$\vdots$$

$$C_n = G_n + P_n C_{n-1}$$

串行进位方式的总延迟时间与字长成正比。假定，一级“与门”、“或门”的延迟时间定为ty，则每一级进位的延迟时间为2ty。在字长为 n 位的情况下，若不考虑 G_i、P_i 的形成时间，从 $C_0 \to C_n$ 的最长延迟时间为 $2n$ty（设 C_0 为加法器最低位的进位，C_n 为加法器最高位的进位）。

2）并行进位方式

并行进位方式所有各位的进位均不依赖于低位的进位，各位的进位可以同时产生。

$$C_1 = G_1 + P_1 C_0$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

$$C_4 = G_4 + P_4 C_3 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$$

$$\vdots$$

这种进位方式是快速的，若不考虑 G_i、P_i 的形成时间，从 $C_0 \to C_n$ 的最长延迟时间仅为2ty，而与字长无关。完全采用并行进位是不现实的。

3）分组先行进位方式

分组先行进位方式又有单级和多级之分。

单级先行进位方式又称为组内并行、组间串行方式。若不考虑 G_i、P_i 的形成时间，从 $C_0 \to C_n$ 的最长延迟时间为 $2m$ty，其中 m 为分组的组数。16位单级先行进位加法器（分为4组，每组4位），从 $C_0 \to C_{16}$ 的最长延迟时间为4×2ty=8ty。

多级先行进位方式又称组内并行、组间并行进位方式。

$$C_4 = G_1^* + P_1^* C_0$$

$$C_8 = G_2^* + P_2^* G_1^* + P_2^* P_1^* C_0$$

$$C_{12} = G_3^* + P_3^* G_2^* + P_3^* P_2^* G_1^* + P_3^* P_2^* P_1^* C_0$$

$$C_{16} = G_4^* + P_4^* G_3^* + P_4^* P_3^* G_2^* + P_4^* P_3^* P_2^* G_1^* + P_4^* P_3^* P_2^* P_1^* C_0$$

其中，

$$G_1^* = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1$$

$$G_2^* = G_8 + P_8 G_7 + P_8 P_7 G_6 + P_8 P_7 P_6 G_5$$

$$G_3^* = G_{12} + P_{12} G_{11} + P_{12} P_{11} G_{10} + P_{12} P_{11} P_{10} G_9$$

$$G_4^* = G_{16} + P_{16} G_{15} + P_{16} P_{15} G_{14} + P_{16} P_{15} P_{14} G_{13}$$

$$P_1^* = P_4 P_3 P_2 P_1$$

$$P_2^* = P_8 P_7 P_6 P_5$$

$$P_3^* = P_{12} P_{11} P_{10} P_9$$

$$P_4^* = P_{16} P_{15} P_{14} P_{13}$$

若不考虑 G_i、P_i 的形成时间，C_0 经过2ty产生第1小组的 C_1、C_2、C_3 以及所有组进位产生函数 G_i^* 和组进位传递函数 P_i^*；再经过2ty，由CLA电路产生 C_4、C_8、C_{12}、C_{16}；再经过2ty后，才能产生第2、3、4小组内的 $C_5 \sim C_7$、$C_9 \sim C_{11}$、$C_{13} \sim C_{15}$，所以最长的进位延迟时间为6ty。

3. 补码加减运算

两个补码表示的数相加,符号位参加运算,且两数和的补码等于两数补码之和,即:

$$[X+Y]_补 = [X]_补 + [Y]_补$$

可以借用加法器来实现减法运算,根据补码加法公式可推出:

$$[X-Y]_补 = [X+(-Y)]_补 = [X]_补 + [-Y]_补$$

从补码减法公式可以看出,只要求得$[-Y]_补$,就可以变减法为加法。已知$[Y]_补$,求$[-Y]_补$的方法是:将$[Y]_补$连同符号位一起求反,末尾加“1”。$[-Y]_补$被称为$[Y]_补$的机器负数,由$[Y]_补$求$[-Y]_补$的过程称为对$[Y]_补$变补(求补),表示为:

$$[-Y]_补 = [[Y]_补]_{变补}$$

要注意将“某数的补码表示”与“变补”这两个概念区分开来。一个正数的补码形式与原码形式相同;一个负数的补码形式是对原码形式除符号位外各位变反,末位加“1”。而变补(求$[-Y]_补$)是对$[Y]_补$包括符号位一起变反(所有的二进制位一起变反),末位加“1”。

4. 符号扩展

在计算机算术运算中,有时必须将采用给定位数表示的数转换成具有更多位数的某种表示形式。例如,某个程序需要将一个8位数与另一个32位数相加。要想得到正确的结果,在将8位数与32位数相加之前,必须将8位数转换成32位数形式,这被称为“符号扩展”。

对于正数的符号扩展非常简单,原有形式数的符号位“0”移动到新形式数的符号位上,新表示形式数的所有附加位都用“0”进行填充。

对于负数的符号扩展方法则根据机器数的不同而有所不同。原码表示负数的符号扩展方法与正数相同,只不过此时符号位为“1”而已。例如,原码表示的8位二进制数10000111,对应的十进制真值是−7,扩展为16位原码形式为1000000000000111。

补码表示负数的扩展方法是:原有形式数的符号位“1”移动到新形式数的符号位上,新表示形式数的所有附加位都用“1”进行填充。例如,补码表示的8位二进制数10000111,对应的十进制真值是−121,扩展为16位补码形式为1111111110000111。

对于补码,符号扩展的另一种理解方式是用符号位来填充附加的高位,即原有符号位保持不变,若为正数则所有附加位都用“0”进行填充,若为负数则所有附加位都用“1”进行填充。

5. 补码的溢出检测方法

设:被操作数为$[X]_补 = X_s, X_1X_2\cdots X_n$。

操作数为$[Y]_补 = Y_s, Y_1Y_2\cdots Y_n$。

其和(差)为$[S]_补 = S_s, S_1S_2\cdots S_n$。

若X、Y异号,不会溢出。若X、Y同号,运算结果为正且大于所能表示的最大正数或者运算结果为负且小于所能表示的最小负数(绝对值最大的负数)时,产生溢出。将两正数相加产生的溢出称为正溢;反之,两负数相加产生的溢出称为负溢。

1) 采用一个符号位

当$X_s=Y_s=0, S_s=1$时,产生正溢;当$X_s=Y_s=1, S_s=0$时,产生负溢。溢出判断条件OVR为:

$$OVR = \overline{X_s}\,\overline{Y_s}S_s + X_sY_s\overline{S_s} = (X_s \oplus S_s)(Y_s \oplus S_s)$$

其逻辑图如图 4-1 所示。

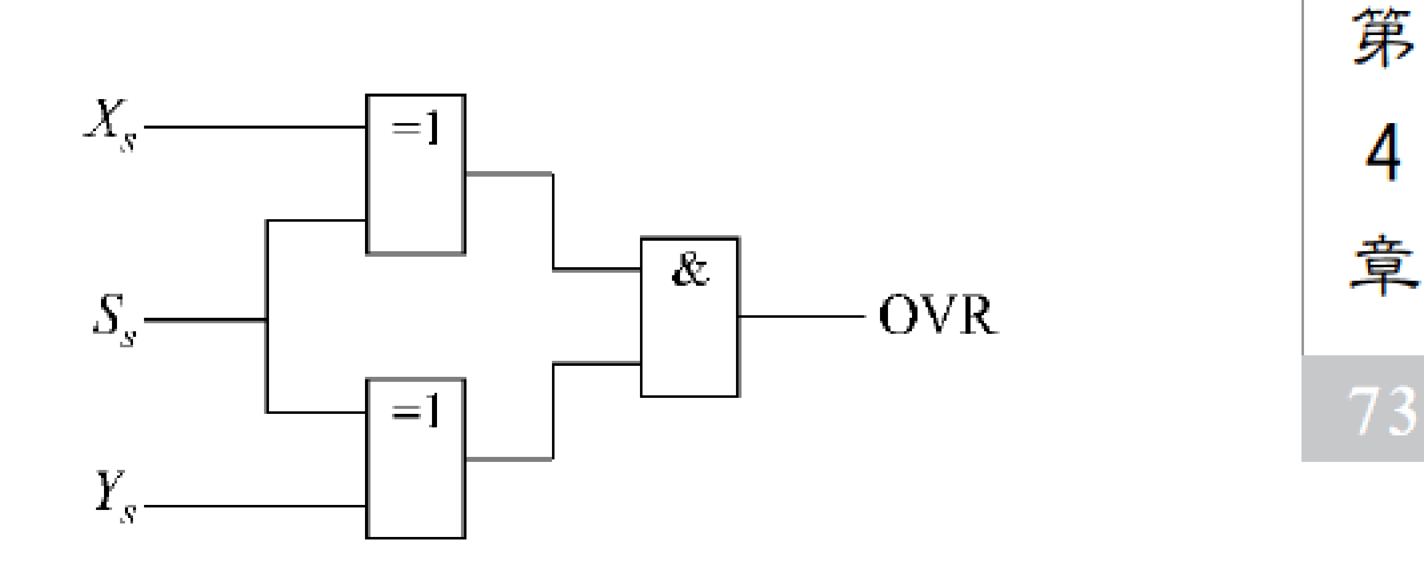


图 4-1　采用一个符号位的溢出检测电路

2）采用进位位

两数运算时，产生的进位为：

$$C_s, C_1 C_2 \cdots C_n$$

其中，C_s 为符号位产生的进位，C_1 为最高数值位产生的进位。

例如：

$$
\begin{array}{r}
[X]_{补} = 0.1010 \\
+\ [Y]_{补} = 0.1001 \\
\hline
[S]_{补} = 1.0011
\end{array}
\qquad C_1 = 1,\ C_s = 0
$$

$$
\begin{array}{r}
[X]_{补} = 1.0001 \\
+\ [Y]_{补} = 1.0111 \\
\hline
[S]_{补} = 0.1000
\end{array}
\qquad C_1 = 0,\ C_s = 1
$$

两正数相加，当最高有效位产生进位（$C_1=1$）而符号位不产生进位（$C_s=0$）时，发生正溢；两负数相加，当最高有效位没有进位（$C_1=0$）而符号位产生进位（$C_s=1$）时，发生负溢。故溢出条件为：

$$\mathrm{OVR} = \overline{C_s} C_1 + C_s \overline{C_1} = C_s \oplus C_1$$

其逻辑图如图 4-2 所示。

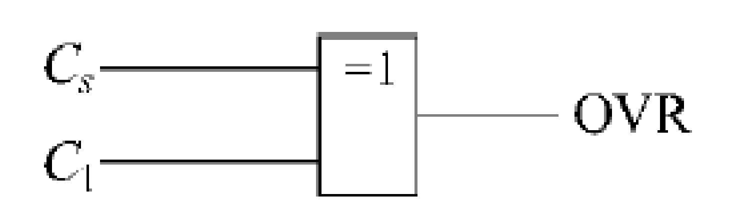


图 4-2　采用进位位的溢出检测电路

3）采用变形补码（双符号位补码）

将被操作数和操作数的符号位均扩充为两位（S_{s1} 和 S_{s2}），双符号位参与运算，如果运算结果如下：

$S_{s1}S_{s2}=00$　正数，无溢出

$S_{s1}S_{s2}=01$　正溢

$S_{s1}S_{s2}=10$　负溢

$S_{s1}S_{s2}=11$　负数，无溢出

当结果的两位符号位的值不一致时，表明产生溢出，溢出条件为：

$$\mathrm{OVR} = S_{s1} \oplus S_{s2}$$

这种溢出检测方法简单，容易实现，只需要在结果的两位符号位上设置一个异或门即可，但是运算器的字长要增加一位。

6．补码定点加减运算的实现

实现补码加减运算的逻辑电路如图 4-3 所示。

图 4-3 中 F 代表一个多位的并行加法器，X 和 Y 是两个寄存器，门 A、B、C 分别是字级的与门和与或门。

当实现补码加法时，需给出 $X \to F$ 和 $Y \to F$ 信号，将$[X]_{补}$和$[Y]_{补}$分别送到 F 的两个输

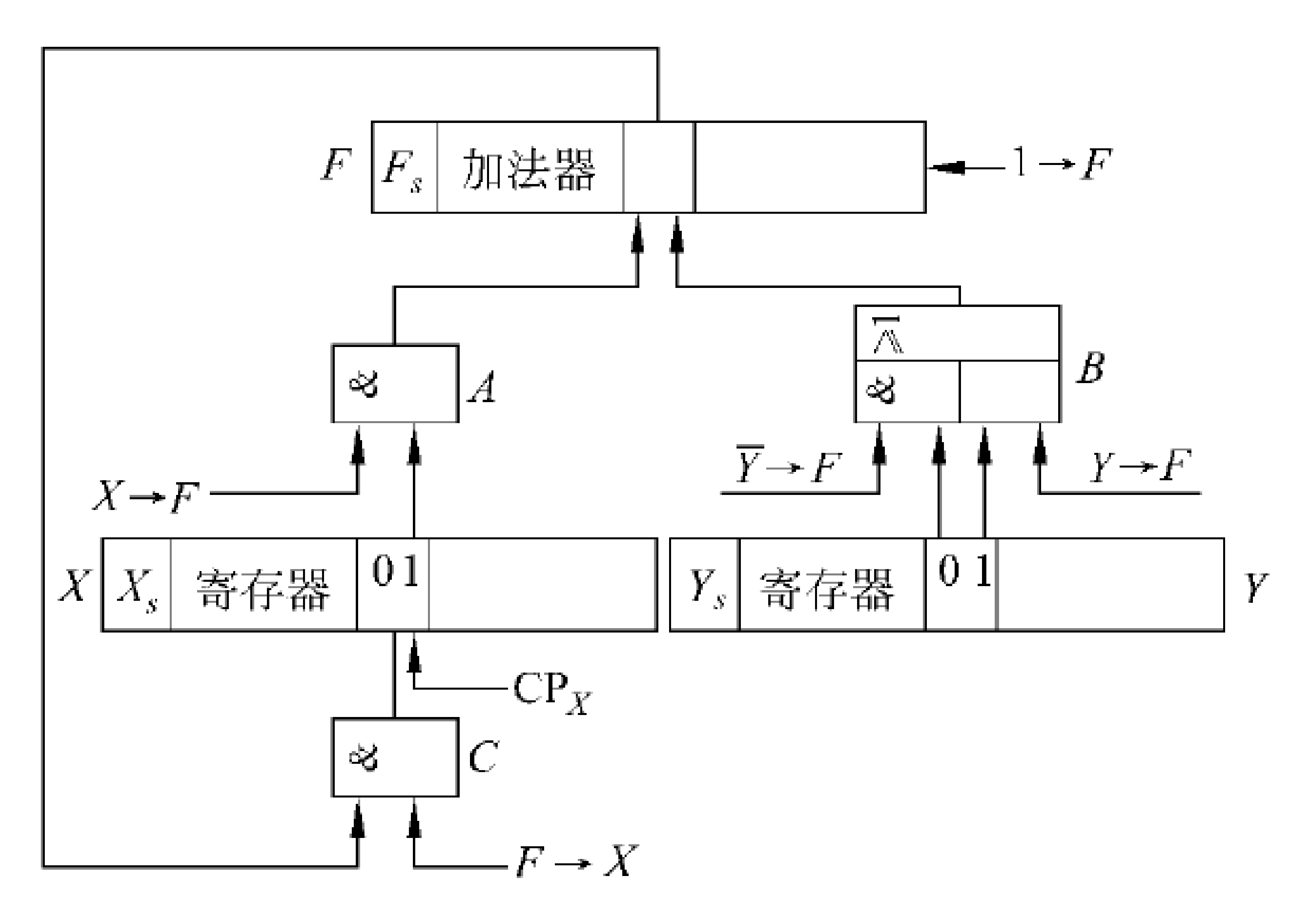


图 4-3 补码加减运算器框图

入端，在加法器 F 中完成$[X]_补+[Y]_补$的加法过程，然后通过 $F\rightarrow X$ 和 CP_X 信号将加法运算结果写入寄存器 X。

当实现补码减法时，需给出 $X\rightarrow F$、$\overline{Y}\rightarrow F$ 和 $1\rightarrow F$ 信号，将$[X]_补$的原变量和$[Y]_补$的反变量分别送到 F 的两个输入端，并使 F 的最低位有进位，在加法器 F 中完成$[X]_补+[-Y]_补$的加法过程，然后通过 $F\rightarrow X$ 和 CP_X 信号将减法运算结果写入寄存器 X。

7. 带符号数的移位规则

算术移位时应该保持数的符号位不变，而数值的大小则要发生变化。左移一位相当于该数乘以 2(不溢出的情况下)；而右移一位相当于该数除以 2(不考虑舍入的情况)。

当机器数为正数时，原码、补码、反码的符号位为“0”，故不论左移或右移，移位后的空出位，一律以“0”补入。

当机器数为负数时，因为 3 种编码的表示方法不同，则移位后的空位必须按不同的规则填补。

负数的原码移位后的空位补“0”。

负数的反码移位后的空位补“1”。这是因为负数反码的各位(符号位除外)均与负数的原码相反，故移位后的空位所补的代码也应与原码相反。

负数的补码左移后的空出位补“0”，右移后的空出位补“1”。这是根据原码和补码的关系推出来的，负数的补码左移时空位出现在低位，其补入代码应与原码相同，即补“0”；右移时空位出现在高位，其补入代码应与反码相同，即补“1”。

8. 补码乘法(Booth 乘法)

在计算机中，可在原有实现加减运算的运算器基础上增加一些逻辑线路以实现乘法运算，即将乘法运算转换成“累加-右移”。

通常，乘法运算需要 3 个寄存器。被乘数$[X]_补$存放在 B 寄存器中；乘数$[Y]_补$存放在 C 寄存器中；A 寄存器用来存放部分积与最后乘积的高位部分，它的初值为 0。运算结束后寄存器 C 中不再保留乘数，改为存放乘积的低位部分。

在乘数的最低位之后增加一位附加位 Y_{n+1}，它的初值为 0，增加附加位不会影响运算结果。

Booth 法(比较法)每次从乘数的最低位开始取两位乘数(Y_{i+1} 和 Y_i)，根据($Y_{i+1}-Y_i$)

的取值(0,－1或＋1)确定是＋0,＋$[-X]_补$,还是＋$[X]_补$,然后右移一位。共进行 $n+1$ 次累加和 n 次右移,便可得到乘积的补码。

注意:由于符号位要参加运算,部分积累加时最高有效位产生的进位可能会侵占符号位,故被乘数和部分积应取双符号位,而乘数只需要一位符号位。

例如,已知 $X=-\frac{21}{32}$,$Y=\frac{26}{32}$,求 $X\times Y$。

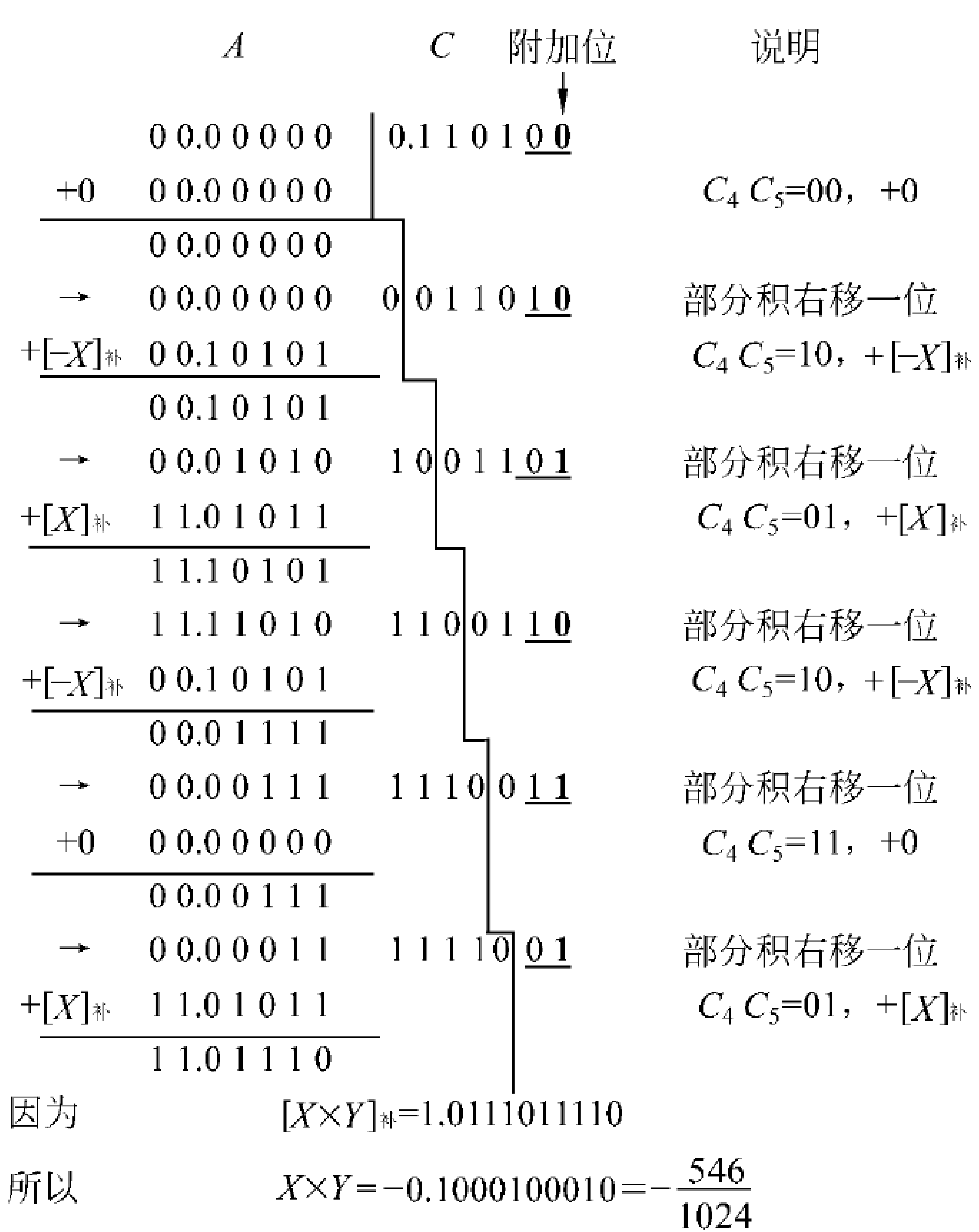
因为,

$$X=-\frac{21}{32}=-21\times 2^{-5}=-0.10101$$

$$Y=\frac{26}{32}=26\times 2^{-5}=0.11010$$

$$[X]_补=1.01011\rightarrow B,\quad [Y]_补=0.11010\rightarrow C,0\rightarrow A$$

$$[-X]_补=0.10101$$



比较法是比较相邻的两位乘数之后决定进行相应的运算,实际每次仅处理一位乘数,所以乘数必须要增加一位附加位,否则相当于对乘数的最低位没有进行处理。

9. 补码除法(加减交替法)

在计算机中,除法运算可转换成"累加-左移"。

通常,除法运算需要3个寄存器。被除数存放在 A 寄存器中;除数存放在 B 寄存器中;C 寄存器用来存放商,它的初值为0。运算过程中 A 寄存器的内容将不断地发生变化,最后

A 寄存器中剩下的是扩大了若干倍的余数。

进行补码除法时符号位参加运算，共需执行 $n+1$ 次累加，n 次左移。补码加减交替法的规则可概括如下：

(1) 若 X 与 Y 同号，则第一次做$[X]_补-[Y]_补$；若 X 与 Y 异号，则第一次做$[X]_补+[Y]_补$。

(2) 若余数与 Y 同号，商“1”，接着做 $2[r_i]_补-[Y]_补$；若余数与 Y 异号，商“0”，接着做 $2[r_i]_补+[Y]_补$。

(3) 将第(2)步操作重复进行 n 次。

(4) 商的最末一位恒置为“1”。

例如，已知 $X=-0.1100$，$Y=-0.1111$，求 $X\div Y$。

$$[X]_补=1.0100\to A,\quad [Y]_补=1.0001\to B,0\to C$$
$$[-Y]_补=0.1111$$

	A	C	说明
	11.0100	0.0000	
$+[-Y]_补$	00.1111		$[X]_补$、$[Y]_补$同号，$+[-Y]_补$
	00.0011	0.000**0**	$[r_i]_补$、$[Y]_补$异号，商 0
←	00.0110		左移一位
$+[Y]_补$	11.0001		$+[Y]_补$
	11.0111	0.00**01**	$[r_i]_补$、$[Y]_补$同号，商 1
←	10.1110		左移一位
$+[-Y]_补$	00.1111		$+[-Y]_补$
	11.1101	0.0**011**	$[r_i]_补$、$[Y]_补$同号，商 1
←	11.1010		左移一位
$+[-Y]_补$	00.1111		$+[-Y]_补$
	00.1001	0.**0110**	$[r_i]_补$、$[Y]_补$异号，商 0
←	01.0010		左移一位
$+[Y]_补$	11.0001		$+[Y]_补$
	00.0011	**0.1101**	末位恒置 1

因为
$$\left[\frac{X}{Y}\right]_补=0.1101+\frac{0.0011\times 2^{-4}}{1.0001}$$

所以
$$\frac{X}{Y}=0.1101+\frac{0.0011\times 2^{-4}}{-0.1111}$$

在补码加减交替除法中采用双符号位进行运算，最左边的符号位是真符。左移时，只要保证真符不在移位中发生变化即可。例如，

00.1××××左移一位为 01.××××0。

11.0××××左移一位为 10.××××0。

部分余数左移一位的结果虽然两个符号位不一致，但这并不表明发生了溢出，只要接着进行一次$+[Y]_补$或$-[Y]_补$运算，以后结果一定能保证两个符号位是一致的。

10. 规格化浮点加减运算

对阶：小阶向大阶看齐。使小阶的阶码增大，则相应的尾数右移，直到两数的阶码相等为止。每右移一位，阶码加1。

尾数加/减：算法同定点补码加/减法。

尾数结果规格化：

当尾数结果为00.0××…×或11.1××…×时，需要使尾数左移以实现规格化，这个过程称为左规。尾数每左移一位，阶码相应减1，直至成为规格化数为止。

当尾数结果为10.×××…×或01.×××…×时，应将尾数右移以实现规格化，这个过程称为右规。尾数每右移一位，阶码相应加1。右规最多只有一次。

因为在尾数规格化时要相应调整其阶码，故有可能出现阶码溢出的情况。阶码溢出则浮点数溢出，即浮点数的溢出情况由阶码的符号决定，若阶码也用双符号位补码表示，当：

$[E_C]_{补}=01,\times\times\times\cdots\times$，表示上溢。此时，浮点数真正溢出，机器需停止运算，做溢出中断处理。

$[E_C]_{补}=10,\times\times\times\cdots\times$，表示下溢。浮点数值趋于零，机器不做溢出处理，而是按机器零处理。

例如，0.110100×2^{-011}，$Y=-0.101110\times2^{-100}$，用补码运算规则求 $X+Y$ 和 $X-Y$。

假设浮点数的阶码和尾数均用补码表示，其中阶码4位(含1位符号位)，尾数7位(含1位符号位)。

$$[X]_{浮}=1101;0.110100$$
$$[Y]_{浮}=1100;1.010010$$

求阶差，对阶

$$\Delta E=E_X-E_Y=1$$

Y 的阶码小，应将 Y 的尾数右移一位，阶码加1。

$$[Y]'_{浮}=1101;1.101001$$

尾数加减：

$$[X]_{尾数}+[Y]'_{尾数}=00.110100+11.101001=00.011101$$
$$[X]_{尾数}-[Y]'_{尾数}=00.110100+00.010111=01.001011$$

结果规格化：

$[X]_{尾数}+[Y]'_{尾数}=00.011101$，执行一次左规处理，$[X+Y]_{补}=1100;0.111010$

$[X]_{尾数}-[Y]'_{尾数}=01.001011$，执行一次右规处理，$[X-Y]_{补}=1110;0.100101$

所以有：

$$X+Y=0.111010\times2^{-100}$$
$$X-Y=0.100101\times2^{-010}$$

此题没有发生溢出。

11. 规格化浮点乘除运算

1) 浮点乘法运算步骤

(1) 判零：检查操作数是否为零，若操作数中有一个为零，乘积必为零，也就勿须做其他操作。若尾数采用原码表示则还需要置结果数符。

(2) 阶码相加：即定点整数加法。如果阶码用移码表示，则在阶码相加后要减去一个

偏移量 2^n。同号时，阶码相加可能会产生溢出，这种溢出在一定情况下会导致整个数据的溢出。阶码相加的步骤放在尾数相乘之前，也是为了在出现溢出后不必进行下面的尾数相乘运算。

(3) 尾数相乘：即定点小数乘法。

(4) 尾数结果规格化：若乘积已是规格化数，勿须再进行规格化操作；若乘积不是规格化数，则需要左规一次。

(5) 溢出判断：溢出分为上溢出和下溢出。乘法运算过程中发生下溢出的可能性有两种：一种情况是阶码和的值太小而发生下溢；另一种情况是，阶码和已经是最小值，乘积尾数左规时阶码仍需减1，从而造成下溢。乘法运算过程中发生上溢出的可能性也有两种：一种情况是阶码和的值太大而发生上溢；另一种情况是，阶码和已经是最大值，乘积尾数右规时阶码仍需加1，从而造成上溢。

2) 浮点除法运算步骤

判零：检查操作数是否为零，如果被除数为零，商必为零，也就勿须做其他操作；若除数为零，则除法为非法操作，应该进行中断处理。商的数符置位规则与乘法相同。

尾数调整：为了使商的尾数是一个定点小数，一定要保证被除数尾数的绝对值小于除数尾数的绝对值，即 $|M_A|<|M_B|$。如果不小于，则令被除数 $|M_A|$ 的尾数右移一位，阶码 E_A 加1。尾数调整最多进行一次。尾数调整的好处是使除法所得到的商必为规格化的定点小数。

阶码相减：即定点整数减法。如果阶码用移码表示，则在阶码相减后要加上一个偏移量 2^n。阶码相减可能产生阶上溢或阶下溢，将阶码相减步骤放在尾数相除运算之前是为了在出现溢出后，不必要进行下面的尾数相除运算。

尾数相除：定点小数除法。

12. 一位十进制整数的加法运算

处理十进制数有两种常见的方法。一种方法是先将输入的十进制数转换为二进制数，在计算机中进行二进制运算，再将运算结果转换为十进制数。这种方法适用于数据量不太多而计算量大的场合。另一种方法是采用二-十进制编码(BCD码)进行十进制运算，这种方法适用于数据量多而计算较简单的场合。

BCD码由4位二进制数表示，按二进制加法规则进行加法。十进制数的进位是10，而4位二进制数的进位是16，为此需要进行必要的十进制校正，才能使该进位正确。不同的BCD码所对应的十进制校正规律是不一样的，因此硬件实现也是不同的。

例如，8421码的加法规则为：

(1) 两个十进制数的8421码相加时，按"逢二进一"的原则进行；

(2) 当和≤9，无须校正；

(3) 当和>9，则+6校正；

(4) 在做+6校正的同时，将产生向上一位的进位。

两个1位十进制数相加，其和不会超过18，考虑低位来的进位，其和最大值是19。表4-1给出了两个1位8421码进行十进制加法运算的所有结果和校正关系。其中，$S_4'S_3'S_2'S_1'$ 代表两个8421码按二进制规则相加的结果，C_4' 代表最高位的进位；$S_4S_3S_2S_1$ 代表8421码的正确结果，C_4 代表向高位的进位。

表 4-1 中需要校正的项可分为如下两部分：

$$C'_4=1$$

$$S'_4S'_3S'_2S'_1=1010\sim1111$$

表 4-1　8421 码的校正关系

十进制数	8421 码					校正前的二进制数					校正关系
	C_4	S_4	S_3	S_2	S_1	C'_4	S'_4	S'_3	S'_2	S'_1	
0～9	0	0	0	0	0	0	0	0	0	0	不校正
			⋮					⋮			
	0	1	0	0	1	0	1	0	0	1	
10	1	0	0	0	0	0	1	0	1	0	+6 校正
11	1	0	0	0	1	0	1	0	1	1	
12	1	0	0	1	0	0	1	1	0	0	
13	1	0	0	1	1	0	1	1	0	1	
14	1	0	1	0	0	0	1	1	1	0	
15	1	0	1	0	1	0	1	1	1	1	
16	1	0	1	1	0	1	0	0	0	0	
17	1	0	1	1	1	1	0	0	0	1	
18	1	1	0	0	0	1	0	0	1	0	
19	1	1	0	0	1	1	0	0	1	1	

用卡诺图化简如图 4-4 所示，可得到校正函数为 $C'_4+S'_4S'_3+S'_4S'_2$，即当 $C'_4+S'_4S'_3+S'_4S'_2=1$ 时，需对和进行+6 校正。

$S'_2S'_1$ \ $S'_4S'_3$	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	0	1	1
10	0	0	1	1

$S'_4S'_3$　$S'_4S'_2$

图 4-4　用卡诺图化简

十进制余 3 码加法规则为：

（1）两个十进制数的余 3 码相加，按"逢二进一"的原则进行；

（2）若其和没有进位，则减 3（即+1101）校正；

（3）若其和有进位，则加 3（即+0011）校正。

最后的校正函数为：

当 $C'_4=0$ 时，−3 校正；当 $C'_4=1$ 时，+3 校正。

为什么会有上述的校正函数呢？试想两个余 3 码分别为：

$$X_{余3码}=X_{8421码}+3$$

$$Y_{余3码}=Y_{8421码}+3$$

当 $X+Y$ 无进位时，结果＝8421 码+6，因此只有减 3，其和才仍为余 3 码。而当 $X+Y$ 有进位时，则说明结果>15，这时两数之和中多余的 6 正好用来跳过 8421 码中的非法码（1010～1111），所以其和已不再余 6 而变成了 8421 码，这时只要再加 3，其和仍为余 3 码。

13. 多功能算术逻辑运算单元 ALU

前述的加法器只能对输入操作数进行加法运算，但在计算机中，还常常要进行逻辑运算和其他的算术运算，多功能算术逻辑运算单元 ALU 不仅能执行两个输入数的多种算术运算，也能执行多种逻辑运算。

1 位算术逻辑运算单元由 1 位全加器和 1 位逻辑运算功能部件组合而成，第 i 位运算单

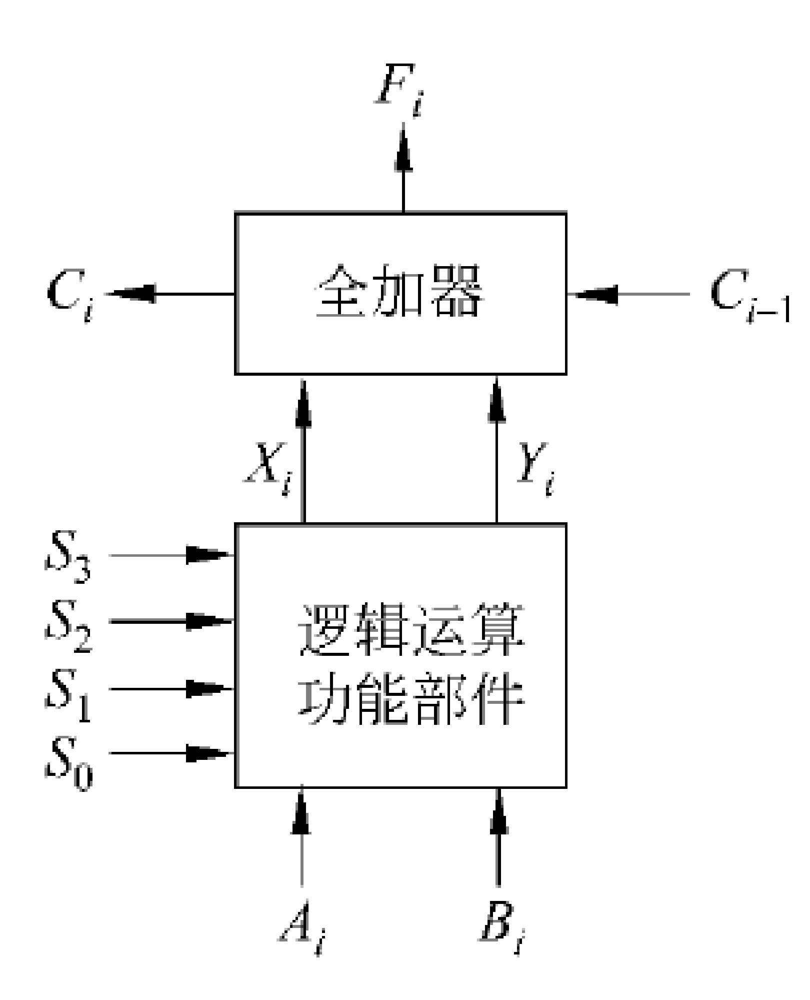


图 4-5 1 位算术逻辑运算单元框图

元的逻辑框图如图 4-5 所示。

算术运算由全加器完成,逻辑运算是在控制信号 $S_3 \sim S_0$ 控制之下由逻辑运算功能部件完成。其中,S_3S_2 用来控制完成 4 种逻辑运算后产生 X_i 进入全加器;S_1S_0 用来控制完成 4 种逻辑运算后产生 Y_i 进入全加器,具体公式如下:

$$X_i = \overline{S_3A_iB_i + S_2A_i\overline{B_i}}$$

$$Y_i = \overline{S_1\overline{B_i} + S_0B_i + A_i}$$

根据上两式,函数 X_i、Y_i 与输入信号 A_i、B_i 的控制关系如表 4-2 所示。

表 4-2 函数 X_i、Y_i 与输入信号 A_i、B_i 的控制关系

S_3S_2	X_i	S_1S_0	Y_i
00	1	00	$\overline{A_i}$
01	$\overline{A_i} + B_i$	01	$\overline{A_i}\,\overline{B_i}$
10	$\overline{A_i} + \overline{B_i}$	10	$\overline{A_i}B_i$
11	$\overline{A_i}$	11	0

将 4 位运算单元(i=0、1、2、3)集成在一个芯片上就构成 74181 芯片,对于算术运算来说,片内的 4 位构成一个小组,小组内采用并行进位方式。

14. ALU 的应用

74181 的 4 位作为一个小组,小组间既可以采用串行进位,也可以采用并行进位。当采用串行进位时,只要把低一片的 C_{n+4} 与高一片的 C_n 相连即可。当采用组间并行进位时,需要增加一片 74182,这是一个先行进位部件。

小组内的并行进位是由 74181 芯片内部完成的,而大组内(小组间)的并行进位是由 74182 芯片完成的,利用 74182 的大组进位产生函数 $\overline{G}$ 和进位传递函数 $\overline{P}$,可进一步实现大组间的并行进位。

显然,如果是 64 位字长的加法器,可分成 16 个小组(每小组包含 4 位),每 4 个小组可构成一个大组,共 4 个大组,于是可用 16 片 74181 和 4 片 74182 构成小组内并行、大组内并行、大组间串行(并—并—串)的 64 位加法器,也可用 16 片 74181 和 5 片 74182 构成采用并—并—并进位方式的 64 位加法器。

4.3 典型例题详解

【例 4.1】 串行加法器和并行加法器有何不同?影响并行加法器的关键因素是什么?设低位来的进位信号为 C_0,请分别按下述两种方式写出 C_4、C_3、C_2、C_1 的逻辑表达式。

(1) 串行进位方式;

(2) 并行进位方式。

解:加法器有串行和并行之分。在串行加法器中,只有一个全加器,数据逐位串行送入

加法器进行运算；并行加法器由多个全加器组成，其位数的多少取决于机器的字长，数据的各位同时运算。

并行加法器可同时对数据的各位相加，但存在着一个加法的最长运算时间问题。并行加法器的最长运算时间主要是由进位信号的传递时间决定的，而每个全加器本身的求和延迟只是次要因素。很明显，提高并行加法器速度的关键是尽量加快进位产生和传递的速度。

进位信号的传递方法有多种。其中，串行进位方式的高一级进位是低一级进位的函数，而并行进位方式的每一级进位都是最低级进位 C_0 的函数。

(1) 串行进位方式

$$C_1 = G_1 + P_1C_0$$
$$C_2 = G_2 + P_2C_1$$
$$C_3 = G_3 + P_3C_2$$
$$C_4 = G_4 + P_4C_3$$

(2) 并行进位方式

$$C_1 = G_1 + P_1C_0$$
$$C_2 = G_2 + P_2G_1 + P_2P_1C_0$$
$$C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_0$$
$$C_4 = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_0$$

【例 4.2】 设操作数信号为 4、3、2、1(最低位信号为 1)。向最低位进位的信号为 C_0，G_i 和 P_i 分别是各位的进位产生函数和进位传递函数。

(1) 完善第 4 位先行进位信号的逻辑表达式。

$$C_4 = G_4 + P_4G_3 + \cdots$$

(2) 基于操作数，试述表达式中各项的实际含义。

解：(1) 第 4 位先行进位信号的逻辑表达式为：

$$C_4 = G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1 + P_4P_3P_2P_1C_0$$

(2) 该表达式中各项的实际含义为：

前 4 项

$$G_4 = A_4B_4$$
$$P_4G_3 = (A_4 \oplus B_4)A_3B_3$$
$$P_4P_3G_2 = (A_4 \oplus B_4)(A_3 \oplus B_3)A_2B_2$$
$$P_4P_3P_2G_1 = (A_4 \oplus B_4)(A_3 \oplus B_3)(A_2 \oplus B_2)A_1B_1$$

这 4 项之和是组进位产生函数，它表示在以上 4 种情况下会产生向高位的进位 C_4。

$$P_4P_3P_2P_1 = (A_4 \oplus B_4)(A_3 \oplus B_3)(A_2 \oplus B_2)(A_1 \oplus B_1)$$

这是组进位传递函数，它表示在满足此条件时，能将最低位的进位 C_0 传递上去。

【例 4.3】 利用 CLA 加法器或 BCLA 加法器以及 CLA 电路设计加法器，要求实现如下功能。

(1) 构建 20 位单级先行进位加法器。

① 使用 5 个 4 位的 CLA 加法器。

② 使用 4 个 5 位的 CLA 加法器。

分别画出连接简图(请特别标明进位信号)。比较这两种方法得到的最长进位延迟时间

有无区别。

(2) 构建20位二级先行进位加法器。

① 使用5个4位的BCLA加法器和1个5位的CLA电路。

② 使用4个5位的BCLA加法器和1个4位的CLA电路。

分别画出连接简图(请特别标明进位信号)。比较这两种方法得到的最长进位延迟时间有无区别。

解:(1) 组内并行,组间串行,每组为4位或5位,共需5组或4组,每一组的进位输出作为下一组的进位输入。①和②两种方式的最长进位延迟时间有区别,前者的进位延迟长于后者,连接简图如图4-6所示。

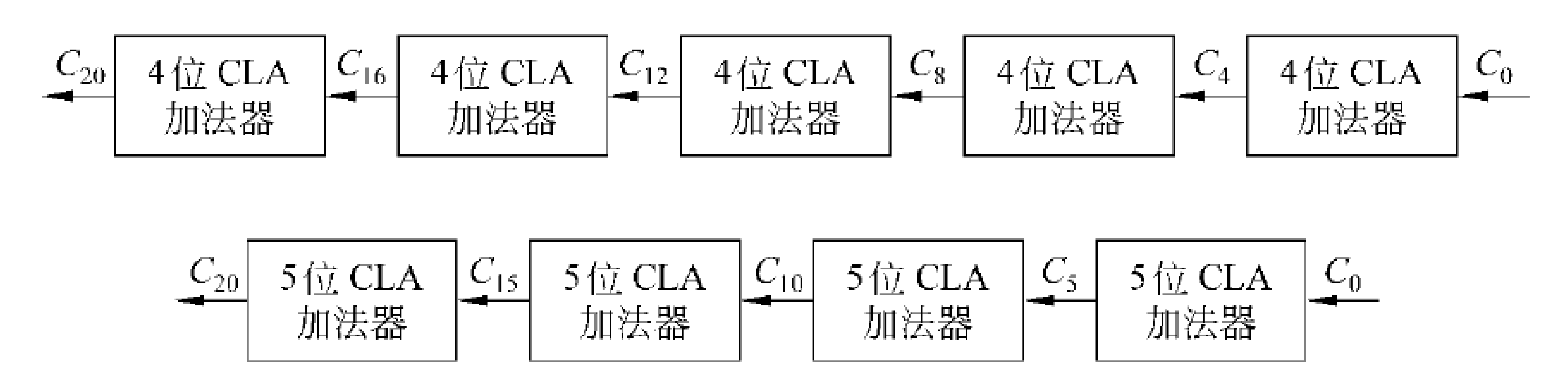


图4-6 20位单级先行进位加法器

(2) 组内并行,组间并行,增加先行进位部件,各组的进位输入来自于先行进位电路。①和②两种方式的最长进位延迟时间无区别,连接简图如图4-7所示。

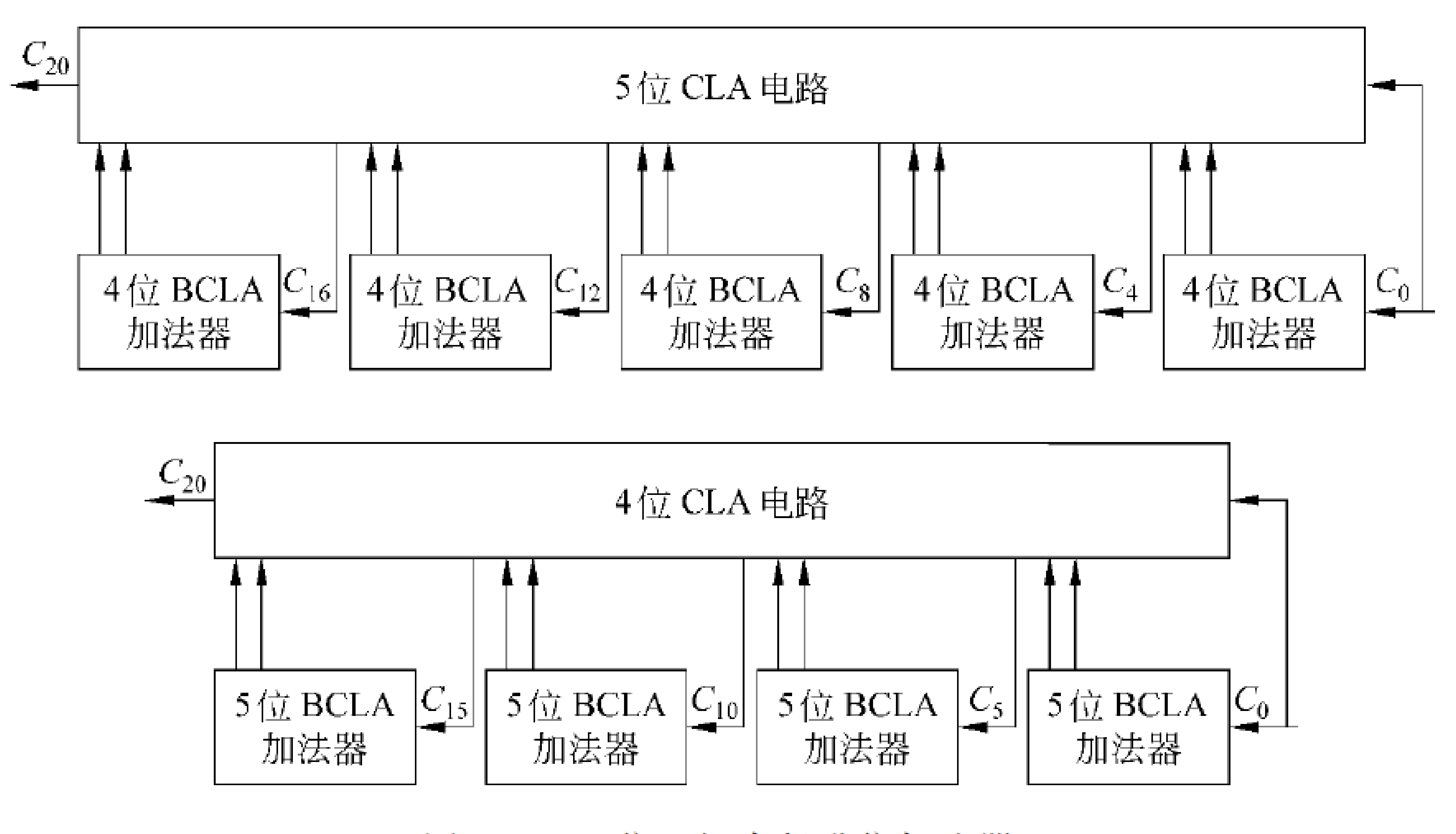


图4-7 20位二级先行进位加法器

【例4.4】 已知X和Y,用变形补码计算$X+Y$和$X-Y$,同时指出运算结果是否溢出?

(1) $X=\frac{27}{32}, Y=\frac{31}{32}$。

(2) $X=\frac{13}{16}, Y=-\frac{11}{16}$。

解:(1) $X=\frac{27}{32}=27\times2^{-5}=0.11011, Y=\frac{31}{32}=31\times2^{-5}=0.11111$

$[X]_{补}=0.11011, [Y]_{补}=0.11111, [-Y]_{补}=1.00001$

$$\begin{array}{lr} & [X]_{补} \quad 00.11011 \\ + & [Y]_{补} \quad 00.11111 \\ \hline & [X+Y]_{补} \quad 01.11010 \end{array}$$

双符号位为01,表示产生正溢。

$$\begin{array}{lr} & [X]_{补} \quad 00.11011 \\ + & [-Y]_{补} \quad 11.00001 \\ \hline & [X-Y]_{补} \quad 11.11100 \end{array}$$

因为$[X-Y]_{补}=1.11100$,所以$X-Y=-0.00100$。

(2) $X=\frac{13}{16}=13\times2^{-4}=0.1101, Y=-\frac{11}{16}=-11\times2^{-4}=-0.1011$

$[X]_{补}=0.1101, [Y]_{补}=1.0101, [-Y]_{补}=0.1011$

$$\begin{array}{lr} & [X]_{补} \quad 00.1101 \\ + & [Y]_{补} \quad 11.0101 \\ \hline & [X+Y]_{补} \quad 00.0010 \end{array}$$

因为$[X+Y]_{补}=0.0010$,所以$X+Y=0.0010$。

$$\begin{array}{lr} & [X]_{补} \quad 00.1101 \\ + & [-Y]_{补} \quad 00.1011 \\ \hline & [X-Y]_{补} \quad 01.1000 \end{array}$$

双符号位为01,表示产生正溢。

【例4.5】 在定点补码加减运算时,产生溢出的条件是什么?试给出几种溢出判断方法(不少于两种,要求写出逻辑表达式,并画出逻辑图)。如果是浮点加减运算,产生溢出的条件又会如何?

解:定点补码加减运算时,产生溢出的条件是:

- 两异号数相加或两同号数相减,不会溢出;
- 两同号数相加或两异号数相减,运算结果为正且大于所能表示的最大正数或者运算结果为负且小于所能表示的最小负数(绝对值最大的负数)时,产生溢出。

判断溢出的方法有以下几种。

(1) 采用一个符号位

溢出判断的逻辑表达式为:

$$溢出 = \overline{X_s}\,\overline{Y_s}S_s + X_sY_s\overline{S_s}$$

逻辑图如图4-1或图4-8(a)所示。

(2) 采用进位位

溢出判断的逻辑表达式为:

$$溢出 = \overline{C_s}C_1 + C_s\overline{C_1} = C_s \oplus C_1$$

逻辑图如图4-2所示。

(3) 采用变形补码(双符号位补码)

溢出判断的逻辑表达式为:

$$溢出 = S_{s1} \oplus S_{s2}$$

逻辑图如图4-8(b)所示。

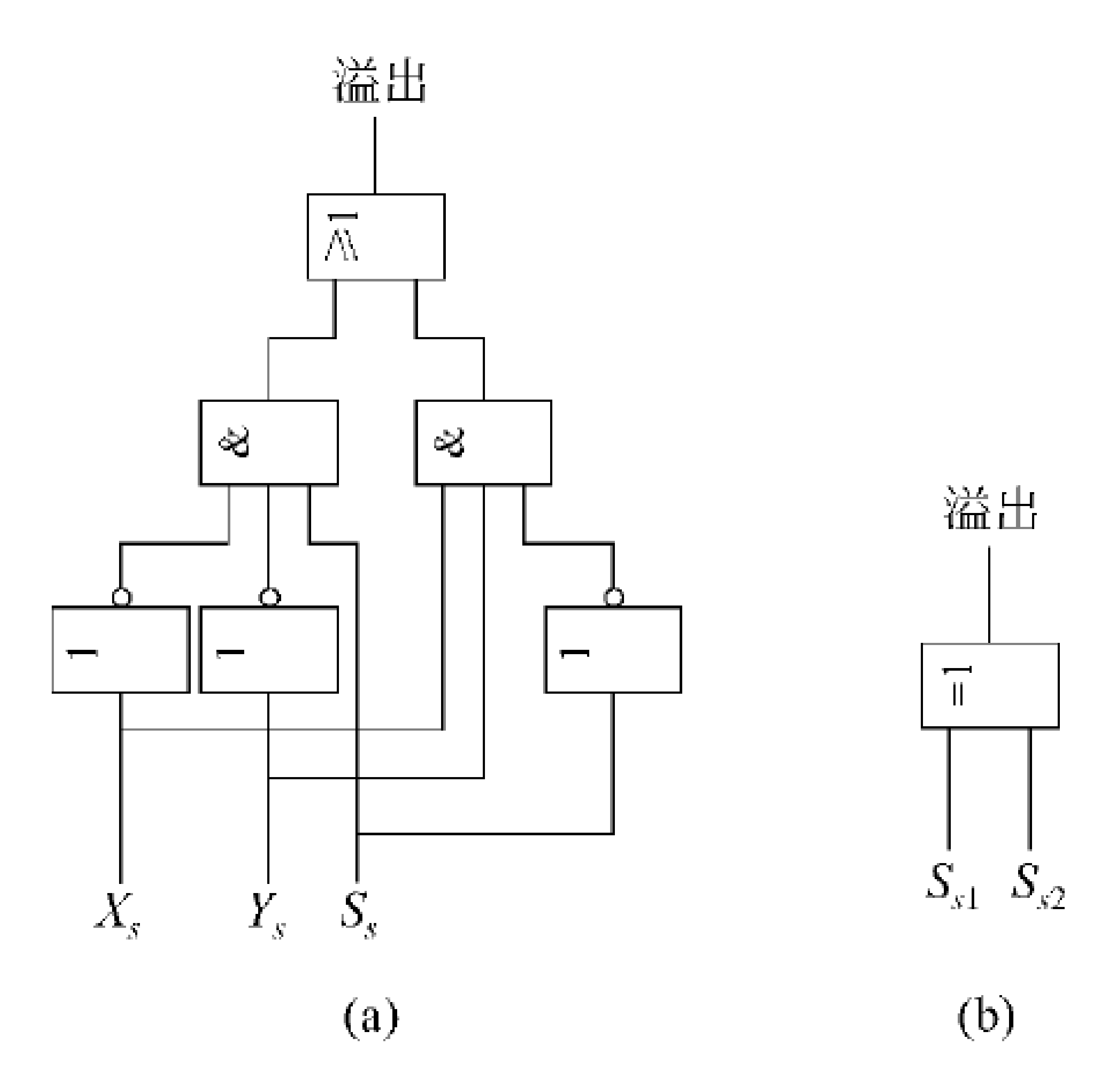


图 4-8　几种溢出判断逻辑电路

【例 4.6】　两个 n 位字长的定点补码数分别放在寄存器 A 和 B 中，A_n 和 B_n 是符号位，用全加器(FA)组成一个 n 位的二进制加、减法器，并列出 $A+B \to A$ 及 $A-B \to A$ 两种运算统一的溢出判断条件逻辑表达式(设用 M 表示方式控制输出信号：当 $M=0$ 时，做加法运算；当 $M=1$ 时，做减法运算)，画出逻辑图。

解：n 位的二进制加、减法器如图 4-9 所示。

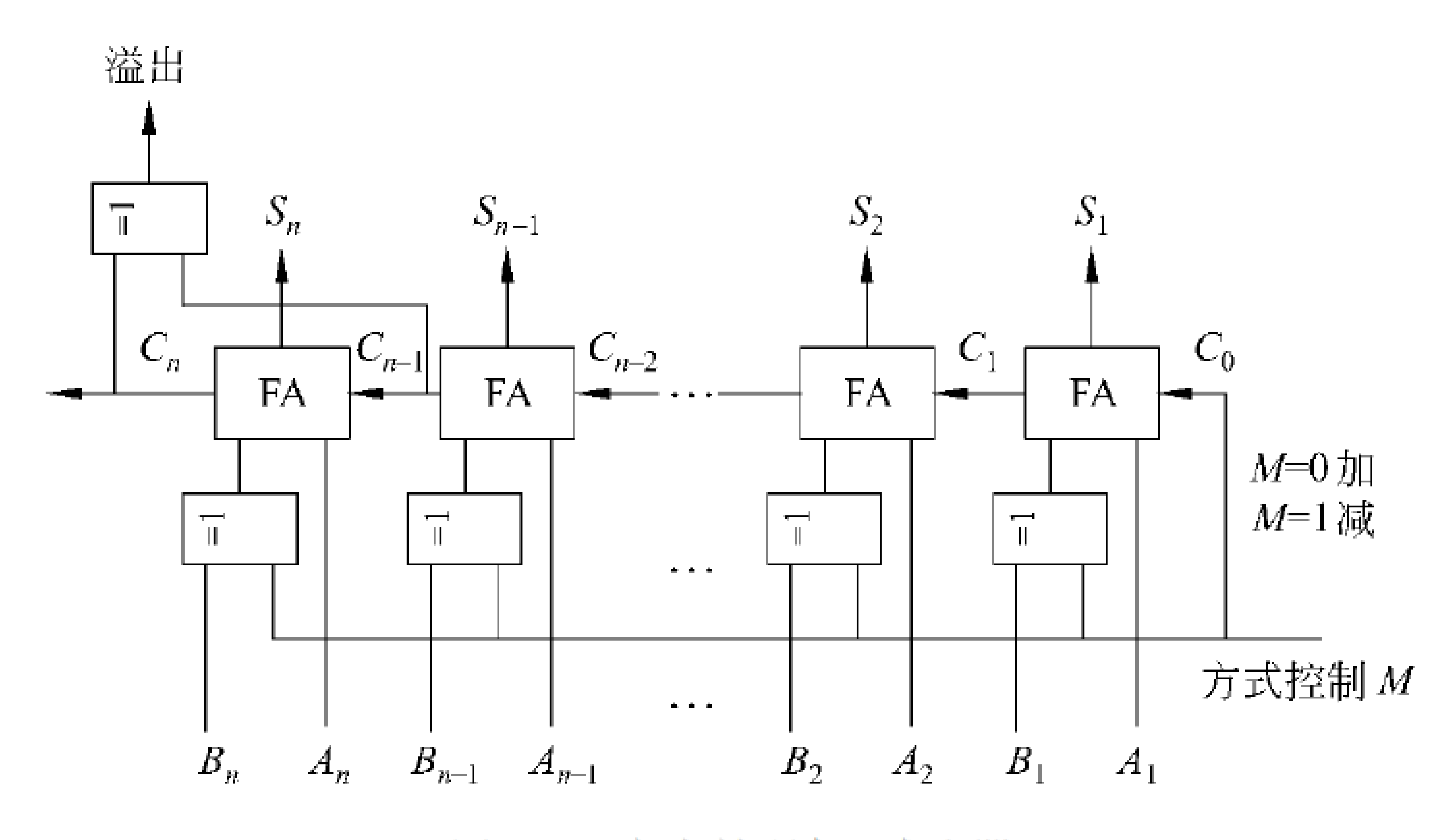


图 4-9　定点补码加、减法器

$$溢出 = C_n \oplus C_{n-1}$$

【例 4.7】　已知 $[X]_补=1.1011000$，$[Y]_补=1.0100110$。计算 $2[X]_补+\frac{1}{2}[Y]_补$。

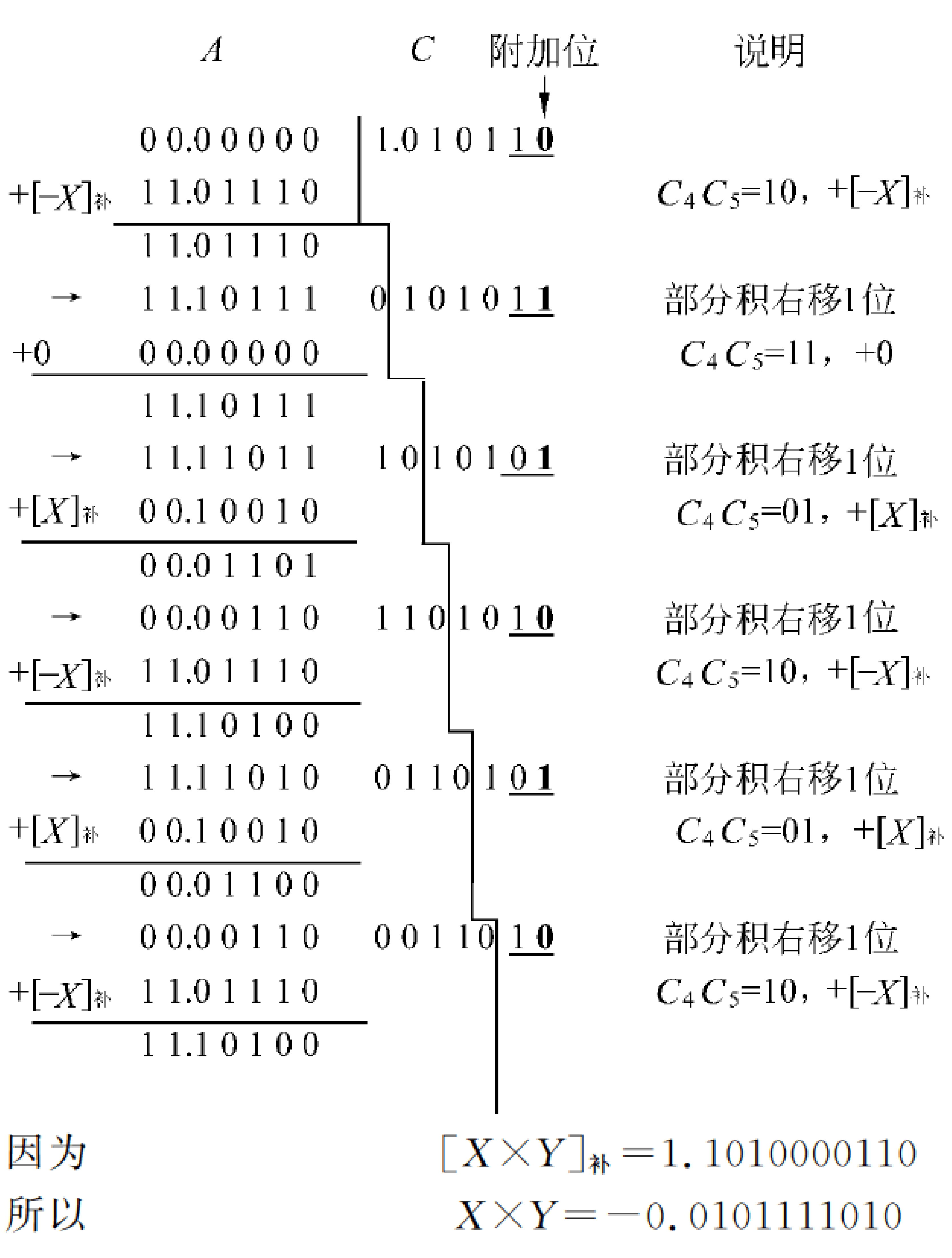
解：为判断溢出采用双符号位，

$$2[X]_补+\frac{1}{2}[Y]_补 = 11.0110000+11.1010011 = 11.0000011$$

【例 4.8】　已知 $X=0.10010$，$Y=-0.10101$，用补码一位乘法计算 $X \times Y$，要写出详细的运算过程。

解：$[X]_补=0.10010 \to B$，$[Y]_补=1.01011 \to C$，$0 \to A$

$[-X]_补=1.01110$



因为 $[X\times Y]_{补}=1.1010000110$

所以 $X\times Y=-0.0101111010$

【例 4.9】 定点补码乘法是否会溢出？若溢出，何时溢出？

解： 若参加运算的两个数均为定点小数，一般不会发生溢出，但当且仅当 $A=B=-1$ 时，$A\times B=1$，这是唯一一种溢出的情况。

【例 4.10】 已知 $X=0.1000$，$Y=-0.1010$，用补码加减交替法求 $\frac{X}{Y}=?$

解： $[X]_{补}=0.1000\to A$，$[Y]_{补}=1.0110\to B$，$0\to C$

$[-Y]_{补}=0.1010$

	A	C	说明
	00.1000	0.0000	
+[Y]补	11.0110		[X]补、[Y]补异号，+[Y]补
	11.1110	0.000**1**	[ri]补、[Y]补同号，商 1
←	11.1100		左移1位
+[−Y]补	00.1010		+[−Y]补
	00.0110	0.00**10**	[ri]补、[Y]补异号，商 0
←	00.1100		左移1位
+[Y]补	11.0110		+[Y]补
	00.0010	0.0**100**	[ri]补、[Y]补异号，商 0
←	00.0100		左移1位
+[Y]补	11.0110		+[Y]补
	11.1010	0.**1001**	[ri]补、[Y]补同号，商 1
←	11.0100		左移1位
+[−Y]补	00.1010		+[−Y]补
	11.1110	**1.0011**	末位恒置 1

因为 $\left[\frac{X}{Y}\right]_{补} = 1.0011 + \frac{1.1110 \times 2^{-4}}{1.0110}$

所以 $\frac{X}{Y} = -0.1101 + \frac{0.0010 \times 2^{-4}}{0.1010}$

【例 4.11】 已知 $X = -7.25, Y = 28.5625$,

(1) 将 X、Y 分别转换成二进制浮点数(阶码占 4 位,尾数占 10 位,各包含一位符号位)。

(2) 用变形补码,求 $X - Y = ?$

解: (1) $X = -7.25 = -111.01\text{B} = -0.11101 \times 2^3$, $Y = 28.5625 = 11100.1001\text{B} = 0.111001001 \times 2^5$

设浮点数的阶码和尾数均采用补码,则有:

$$[X]_{浮} = 0011;1.000110000$$

$$[Y]_{浮} = 0101;0.111001001$$

(2) 因为 X 的阶码小,所以 X 的尾数右移 2 位,阶码加 2,则有:

$$[X]'_{浮} = 0101;1.110001100$$

对阶之后,尾数相减,

$$\begin{array}{r} 11.110001100 \\ +11.000110111 \\ \hline 10.111000011 \end{array}$$

需右规处理,阶码加 1。

因为 $[X - Y]_{浮} = 0110;1.011100001$

所以 $X - Y = -0.100011111 \times 2^{110} = -100011.111 = -35.875$

结果出现误差的原因是因为有舍入误差。

【例 4.12】 有两个浮点数 $X = 2^{10} \times (0.101)$, $Y = 2^{01} \times (-0.111)$,设阶符 1 位,阶码 2 位,数符 1 位,尾数 3 位,用补码运算规则计算 $X + Y$ 的值。

解: $[X]_{浮} = 010;0.101$, $[Y]_{浮} = 001;1.001$

对阶:因为 Y 的阶码小,所以 Y 的尾数右移 1 位,阶码加 1,则有

$$[Y]'_{浮} = 010;1.100$$

尾数相加:

$$[X]_{尾数} + [Y]'_{尾数} = 00.101 + 11.100 = 00.001$$

结果规格化:尾数需要进行左规,结果尾数左移 2 位,阶码减 2。

溢出判断:未发生溢出。

因为 $[X + Y]_{浮} = 000;0.100$

所以 $X + Y = 2^{00} \times (0.100)$

【例 4.13】 利用 4 位 MSI 二进制加法器、全加器、半加器和必要的逻辑门电路,设计2 位并行十进制加法器电路,其输入的十进制为余 3 码。要求和为 8421 码形式,4 位 MSI 二进制加法器的逻辑框图如图 4-10 所示。

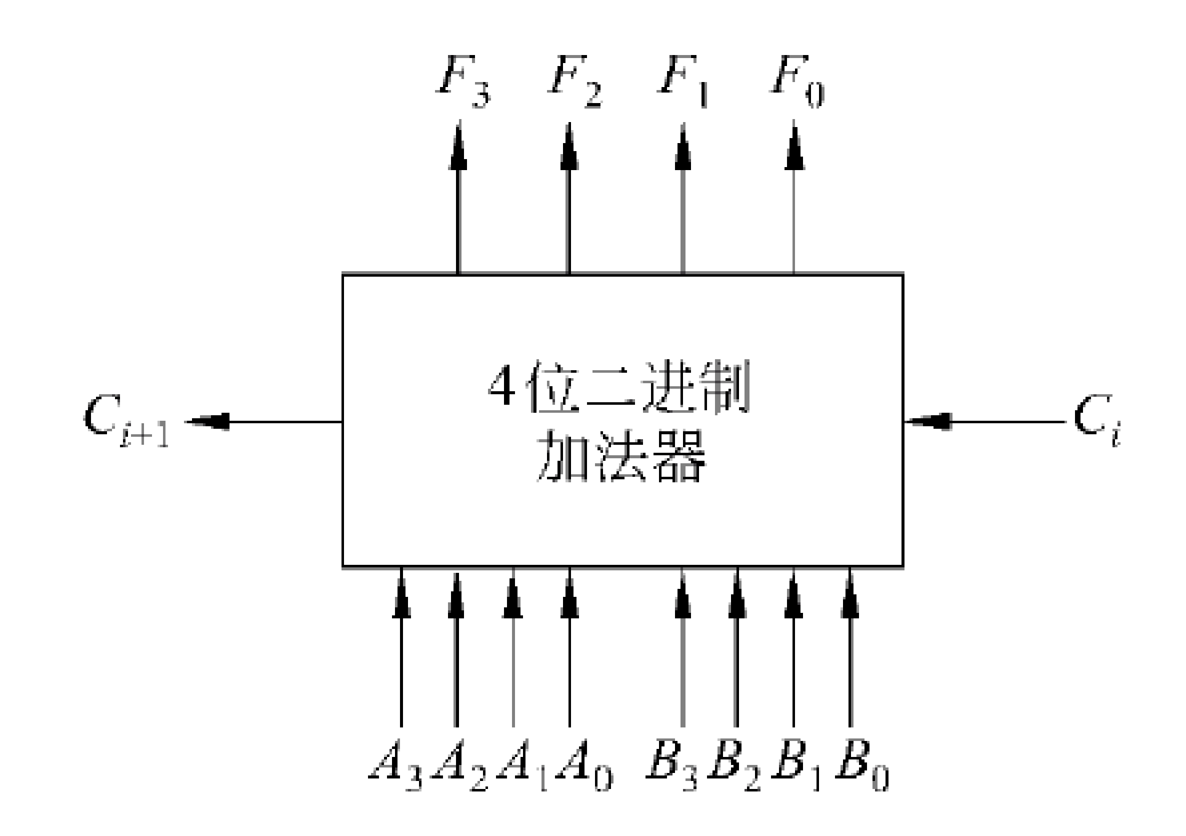


图 4-10 4 位 MSI 二进制加法器的逻辑框图

解：输入为余3码，和为8421码的十进制加法器的校正关系如表4-3所示。

表4-3　十进制加法器的校正关系

十进制数	输入(余3码)	校正前输出	校　　正
0	0 0011	0 0110	
1	0 0100	0 0111	
2	0 0101	0 1000	
3	0 0110	0 1001	无进位
4	0 0111	0 1010	−6(+1010)
5	0 1000	0 1011	
6	0 1001	0 1100	
7	0 1010	0 1101	
8	0 1011	0 1110	
9	0 1100	0 1111	
10	1 0011	1 0000	
11	1 0100	1 0001	
12	1 0101	1 0010	
13	1 0110	1 0011	有进位
14	1 0111	1 0100	不修正
15	1 1000	1 0101	
16	1 1001	1 0110	
17	1 1010	1 0111	
18	1 1011	1 1000	
19	1 1100	1 1001	

从表4-3中分析得出，本位无进位时，−6校正；本位有进位时，无须校正。2位并行十进制加法器电路如图4-11所示。

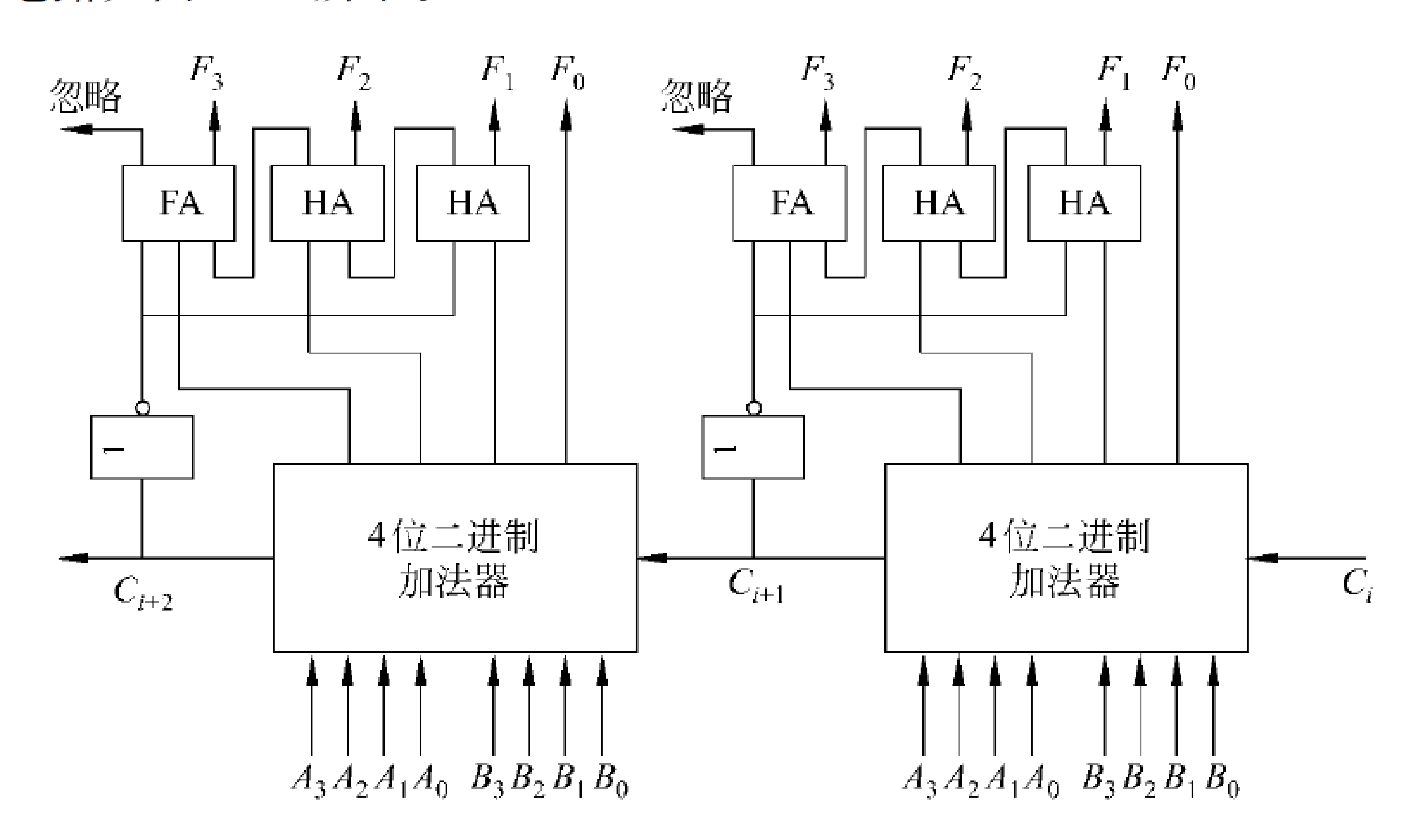


图4-11　2位并行十进制加法器电路

【例4.14】 试用全加器、异或门和与门设计一个加法器网络，它可将2位8421码的十进制整数转换成二进制整数。(提示：先找出2位二进制数的8421码与其对应的二进制数各位的关系)

解：设 2 位十进制数的 8421 码由 A_2A_1 组成，其中 $A_2=a_7a_6a_5a_4$，它是 8421 码的十位数，$A_1=a_3a_2a_1a_0$，它是 8421 码的个位数。

转换后的二进制码为 $b_6b_5b_4b_3b_2b_1b_0$，则有：

$$\begin{aligned} b_6b_5b_4b_3b_2b_1b_0 &= A_2\times 1010+A_1=(a_7a_6a_5a_4)\times 1010+a_3a_2a_1a_0 \\ &=(a_7a_6a_5a_4)\times 1000+(a_7a_6a_5a_4)\times 10+a_3a_2a_1a_0 \end{aligned}$$

即

$$\begin{array}{rccccccc} & a_7 & a_6 & a_5 & a_4 & 0 & 0 & 0 \\ & & & a_7 & a_6 & a_5 & a_4 & 0 \\ + & & & & a_3 & a_2 & a_1 & a_0 \\ \hline & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \end{array}$$

除最后一列不需要做加法运算外，第 3 列至第 5 列需要全加器，其余各列由异或门、与门等构成，加法器网络如图 4-12 所示。

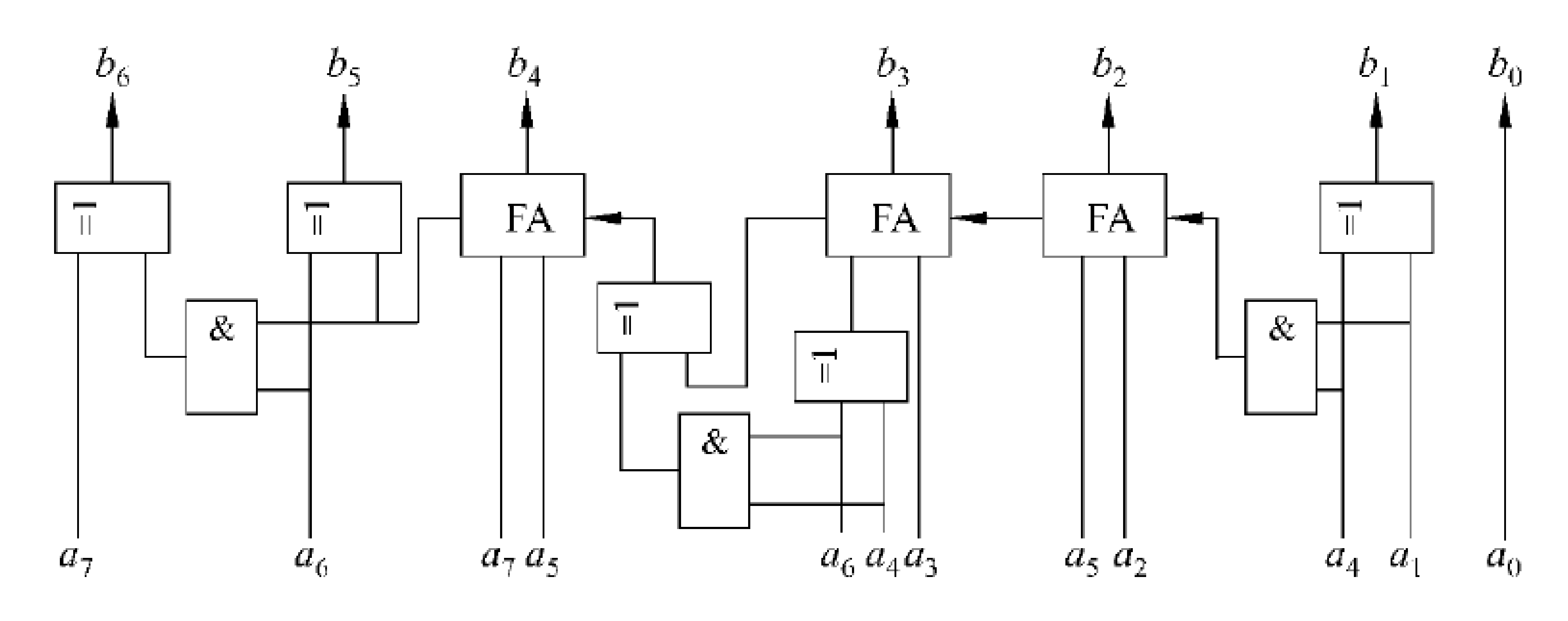


图 4-12　加法器网络

【例 4.15】 利用 74181 和 74182 芯片设计如下 3 种方案的 32 位 ALU。

(1) 行波进位方案；

(2) 二级先行进位方案；

(3) 三级先行进位方案。

解：74181 是 4 位的 ALU 芯片，74182 是先行进位芯片，74181 与 74182 配合使用，可实现各种不同结构的 32 位 ALU。

(1) 行波进位方案

该方案仅使用 8 片 74181 芯片，用前一级芯片的进位输出端作为下一级芯片进位输入端，片内先行进位，片间串行进位。运算速度最慢，如图 4-13 所示。

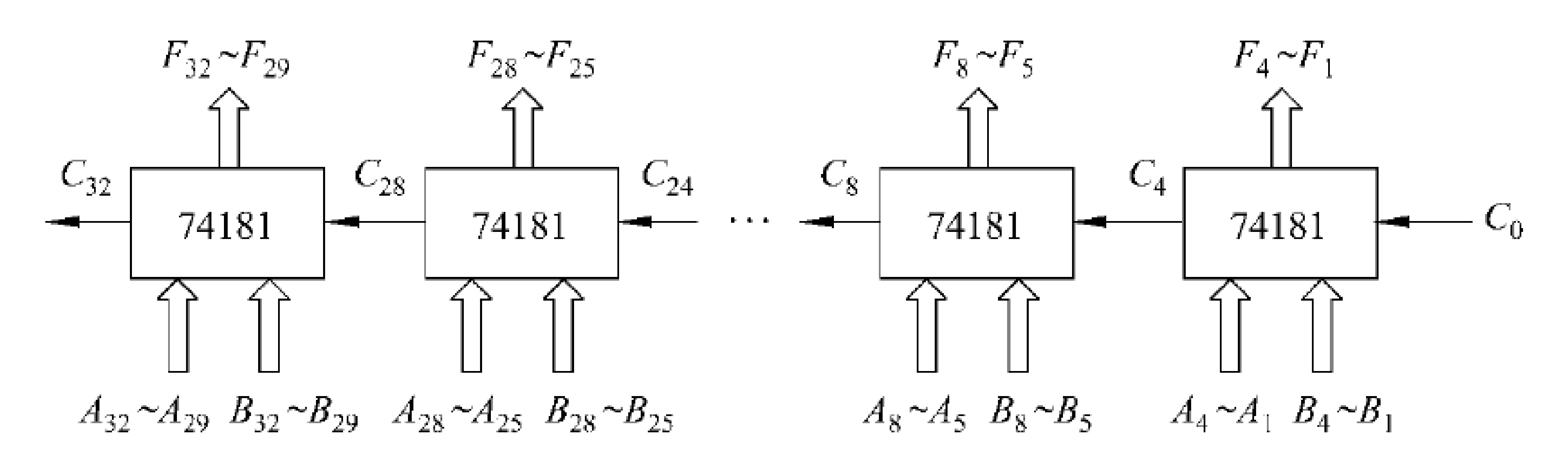


图 4-13　32 位行波进位方案的 ALU

(2) 二级先行进位方案

该方案需使用 8 片 74181 芯片，2 片 74182。每 4 片 74181 为一组，使用 1 片 74182，可

实现 4 片 74181 之间的第二级先行进位。最后组成一个小组内并行、大组内并行、大组间串行的 32 位加法器，运算速度较快，如图 4-14 所示。

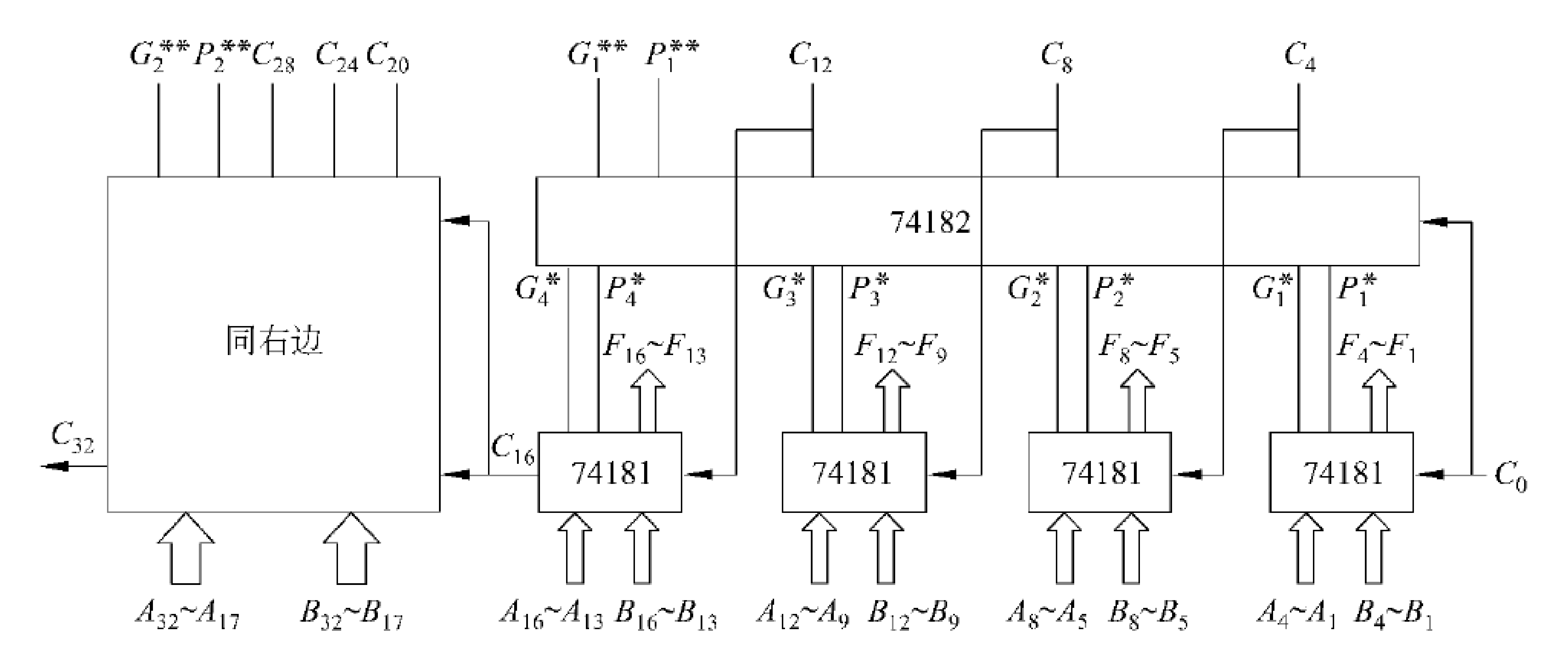


图 4-14　32 位二级先行进位方案的 ALU

（3）三级先行进位方案

该方案需使用 8 片 74181 芯片，3 片 74182。多用一片 74182，以实现第三级先行进位。最后组成一个小组内并行、大组内并行、大组间并行的 32 位加法器，运算速度最快。如图 4-15 所示。

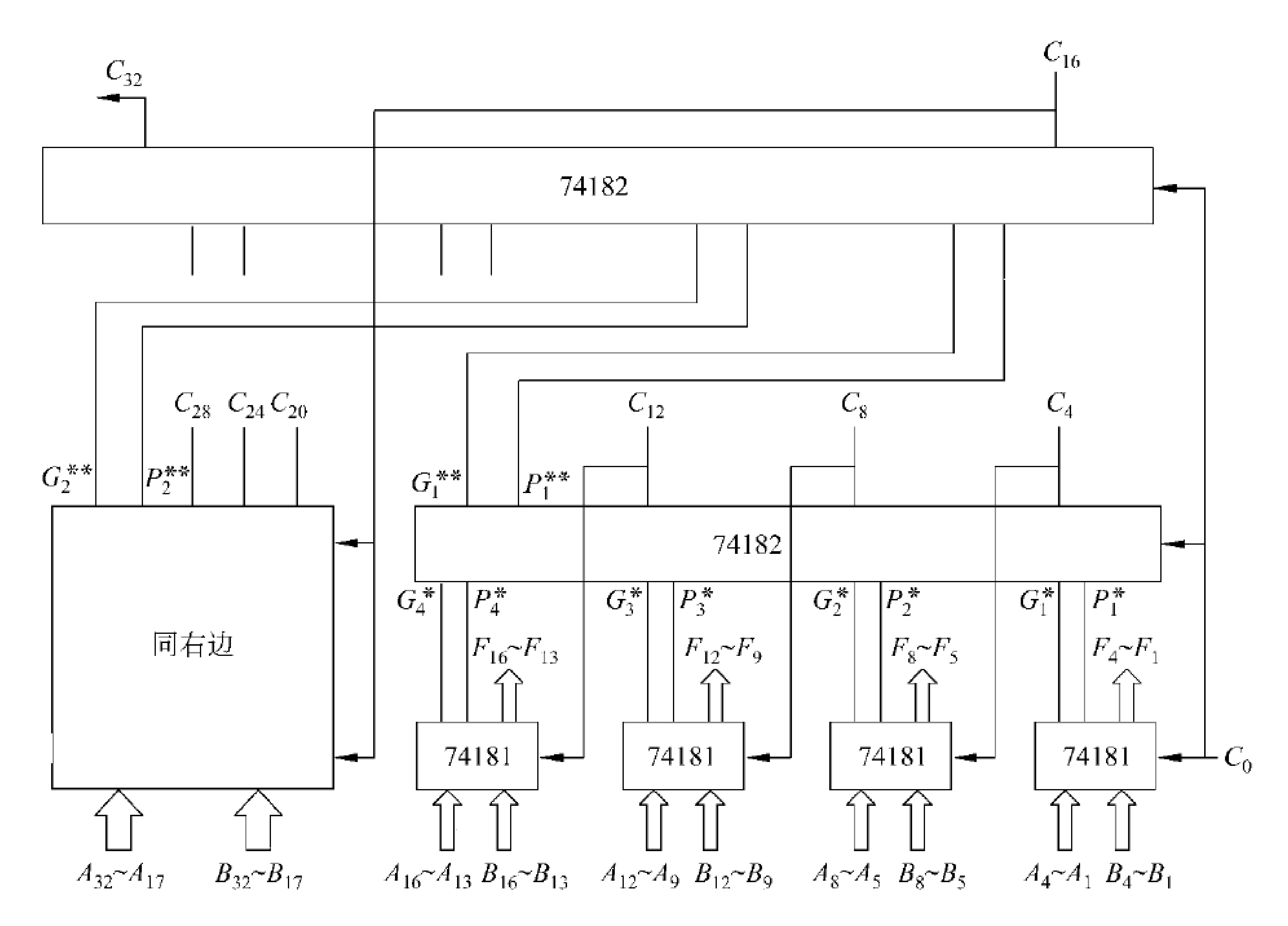


图 4-15　32 位三级先行进位方案的 ALU

*【例 4.16】　一个 C 语言程序在一台 32 位机器上运行。程序中定义了 3 个变量 x、y 和 z，其中 x 和 z 为 int 型，y 为 short 型。当 $x=127$，$y=-9$ 时，执行赋值语句 $z=x+y$ 后，x、y 和 z 的值分别是________。

A. x=0000007FH,y=FFF9H,z=00000076H

B. x=0000007FH,y=FFF9H,z=FFFF0076H

C. x=0000007FH,y=FFF7H,z=FFFF0076H

D. x=0000007FH,y=FFF7H,z=00000076H

解:D。

分析:当两个不同长度的数据,要想通过算术运算得到正确的结果,必须将短字长数据转换成长字长数据,这被称为"符号扩展"。x 和 z 为 int 型,数据长 32 位,y 为 short 型,数据长 16 位,均用补码表示。因为 x=127D=1111111B,y=−9D=−1001B,所以有 x=0000007FH,y=FFF7H。执行赋值语句 $z=x+y$,y 需要扩展符号位之后,再与 x 相加,$z=x+y$=0000007F+FFFFFFF7H=00000076H。

*【**例 4.17**】 浮点数加、减运算一般包括对阶、尾数运算、规格化、舍入和判断溢出等步骤。设浮点数的阶码和尾数均采用补码表示,并且位数分别为 5 位和 7 位(均含 2 位符号位)。若有两个数 $X=2^7\times29/32$,$Y=2^5\times5/8$,则用浮点加法计算 $X+Y$ 的最终结果是________。

A. 00111 1100010　　　　B. 00111 0100010

C. 01000 0010001　　　　D. 发生溢出

解:D。

分析:浮点数加、减运算一般包括对阶、尾数运算、规格化、舍入和判断溢出等步骤。第一步,对阶:第一个数 $X=2^7\times29/32$,浮点数格式为 00111 0011101,第二个数 $Y=2^5\times5/8$,浮点数格式 00101 0010100。对阶原则是小阶向大阶看齐,M_Y 右移 2 位,E_Y+2,浮点数格式为 00111 0000101。第二步,尾数相加:$M_Z=M_X+M_Y$=0100010,浮点数格式为00111 0100010。第三步,结果规格化:尾数需要进行一次右规,才能变成规格化数,M_Z 右移 1 位,E_Z+1,浮点数格式为 01000 0010001。第四步,判断溢出:由于阶码符号位不同,所以发生溢出。

此题很容易误选为 B、C。这是因为 B、C 两个选项本身并没有计算错误,只是它们不是最终结果,B 选项少了第三步和第四步,C 选项少了第四步。

*【**例 4.18**】 假定有 4 个整数用 8 位补码分别表示为 r_1=FEH,r_2=F2H,r_3=90H,r_4=F8H。若将运算结果存放在一个 8 位寄存器中,则下列运算会发生溢出的是________。

A. $r_1\times r_2$　　B. $r_2\times r_3$　　C. $r_1\times r_4$　　D. $r_2\times r_4$

解:B。

分析:用补码表示时 8 位寄存器所能表示的整数范围为−128~+127。现在 4 个整数都是负数,$r_1=-2$,$r_2=-14$,$r_3=-112$,$r_4=-8$,只有 $r_2\times r_3=1568$,结果溢出,其余 3 个算式结果都未超过 127,不发生溢出。

这道题表面上看是在考查定点乘法运算,实际上是在考查 8 位定点整数的表示范围。

*【**例 4.19**】 某字长为 8 位的计算机中,已知整型变量 x、y 的机器数分别为$[x]_{补}$=1 0000100,$[y]_{补}$=1 0110000。若整型变量 $z=2\times x+y/2$,则 z 的机器数为________。

A. 1 1000000　　B. 0 0100100　　C. 1 0101010　　D. 溢出

解:A。

分析:求 $z=2\times x+y/2$,就是将 x 左移一位,y 右移一位,然后再相加。由于$[x]_{补}$=

11110100，则 $2[x]_{补}=11101000$；$[y]_{补}=10110000$，则 $1/2[y]_{补}=11011000$，两者相加结果为 11000000。

*【例 4.20】 若 $x=103$，$y=-25$，则下列表达式采用 8 位定点补码运算实现时，会发生溢出的是________。

A. $x+y$　　B. $-x+y$　　C. $x-y$　　D. $-x-y$

解：C。

分析：参加运算的两个数一正一负，两个异号的数只有在做减法运算才有可能发生溢出，因此选项 A 和 D 可以排除。选项 B 的结果为−128，选项 C 的结果为+128，8 位定点补码的表示范围为−128 ～+127，所以只有选项 C 发生溢出。

*【例 4.21】 假定在一个 8 位字长的计算机中运行下列类 C 程序段：

```
unsigned int  x=134;
unsigned int  y=246;
int  m=x;
int  n=y;
unsigned int  z1=x-y;
unsigned int  z2=x+y;
int  k1=m-n;
int  k2=m+n;
```

若编译器编译时将 8 个 8 位寄存器 R1～R8 分别分配给变量 x、y、m、n、$z1$、$z2$、$k1$ 和 $k2$。请回答下列问题。（提示：带符号整数用补码表示）

（1）执行上述程序段后，寄存器 R1、R5 和 R6 的内容分别是什么？（用十六进制表示）

（2）执行上述程序段后，变量 m 和 $k1$ 的值分别是多少？（用十进制表示）

（3）上述程序段涉及带符号整数加/减、无符号整数加/减运算，这 4 种运算能否利用同一个加法器及辅助电路实现？简述理由。

（4）计算机内部如何判断带符号整数加/减运算的结果是否发生溢出？上述程序段中，哪些带符号整数运算语句的执行结果会发生溢出？

解：（1）无符号整数运算，$R1=x=10000110B=86H$、$R5=x-y=10010000B=90H$、$R6=x+y=01111100B=7CH$。

（2）带符号整数运算，$m=-122$，$k1=x-y=-112$。

（3）4 种运算可以利用同一个加法器及辅助电路实现，因为所做的运算是相同的，只是带符号整数加/减运算结果要考虑溢出，无符号整数加/减运算结果不考虑溢出。

（4）判断溢出的方法有 3 种：一位符号位、进位位和双符号位。上述程序段中只有 int $k2=m+n$ 语句会发生溢出，因为两个带符号整数均为负数，它们相加之后，结果小于 8 位二进制所能表示的最负的数。

分析：本题涉及无符号数和带符号数的加/减运算、二进制加法器以及定点数溢出判断等问题。

在这段类 C 语言程序段中，前两个数据为无符号整数，后两个数据为带符号整数。前两条运算语句为无符号整数运算，后两条运算语句为带符号整数运算。由于两个无符号整数均大于 128，表明其最高位为“1”；如果转换为带符号整数，则两个数均为负数。首先将两

个十进制数转换成8位二进制数,$x=134=10000110\text{B}$,$y=246=11110110\text{B}$,然后进行运算。

4.4 同步测试习题及解答

4.4.1 同步测试习题

一、填空题

1. 影响并行加法器速度的关键因素是________。

2. A、B 均为8位二进制数,$A=\text{F0H}$,$B=\text{E0H}$,则 $A+B=$________,$A-B=$________。

3. 已知某数的补码为11110101,算术左移1位后得________,算术右移1位后得________。

4. 向左规格化的规则为尾数________,阶码________。

5. 运算器的基本功能是实现________运算和________运算。

二、选择题

1. 在串行进位的并行加法器中,影响加法器运算速度的关键因素是________。
   A. 门电路的级延迟　　B. 元器件速度
   C. 进位传递延迟　　D. 各位加法器速度的不同

2. 并行加法器中每一位的进位产生函数 G_i 为________。
   A. $A_i \cdot B_i$　　B. $A_i \oplus B_i$　　C. $A_i \oplus B_i \oplus C_{i-1}$　　D. $A_i+B_i+C_{i-1}$

3. 补码加/减法是指________。
   A. 操作数用补码表示,两尾数相加/减,符号位单独处理
   B. 操作数用补码表示,符号位和尾数一起参加运算,结果的符号与加/减数相同
   C. 操作数用补码表示,连同符号位直接相加,减某数用加某数的机器负数代替,结果的符号在运算中形成
   D. 操作数用补码表示,由数符决定两尾数的操作,符号位单独处理

4. 两个补码数相加,采用1位符号位,当________时,表示结果溢出。
   A. 符号位有进位
   B. 符号位进位和最高数位进位异或结果为0
   C. 符号位为1
   D. 符号位进位和最高数位进位异或结果为1

5. 在双符号位判断溢出的方案中,出现正溢出时,双符号位应当为________。
   A. 00　　B. 01　　C. 10　　D. 11

6. 在定点机中执行算术运算时会产生溢出,其原因是________。
   A. 主存容量不够　　B. 操作数过大
   C. 操作数地址过大　　D. 运算结果无法表示

7. 当定点运算发生溢出时,应进行________。
   A. 向左规格化　　B. 向右规格化

C. 发出出错信息　　D. 舍入处理

8. 8 位补码 10010011 等值扩展为 16 位后，其机器数为________。

A. 1111111110010011　　B. 0000000010010011

C. 1000000010010011　　D. 1111111101101101

9. 将用 8 位二进制补码表示的十进制数−121，扩展成 16 位二进制补码，结果用十六进制表示为________。

A. 0087H　　B. FF87H　　C. 8079H　　D. FFF9H

10. 已知 $\left[\frac{X}{2}\right]_{补}$=C6H，计算机的机器字长为 8 位二进制编码，则$[X]_{补}$=________。

A. 8CH　　B. 18H　　C. E3H　　D. F1H

11. 对于二进制数，若小数点左移 1 位则数值________，若小数点右移 1 位则数值________。

A. 扩大一倍，扩大一倍　　B. 扩大一倍，缩小一半

C. 缩小一半，扩大一倍　　D. 缩小一半，缩小一半

12. X、Y 为定点二进制数，其格式为 1 位符号位，n 位数值位。若采用 Booth 补码一位乘法实现乘法运算，则最多需要做加法运算的次数是________。

A. $n-1$　　B. n　　C. $n+1$　　D. $n+2$

13. 原码加减交替除法又称为不恢复余数法，因此________。

A. 不存在恢复余数的操作

B. 当某一步运算不够减时，做恢复余数的操作

C. 仅当最后一步余数为负时，做恢复余数的操作

D. 当某一步余数为负时，做恢复余数的操作

14. 在加法器、寄存器的基础上增加部分控制电路实现乘除法时，用 B 寄存器存放________。

A. 被乘数和被除数　　B. 被乘数和除数

C. 乘数和被除数　　D. 乘数和除数

15. 若浮点数用补码表示，判断运算结果是否是规格化数的方法是________。

A. 阶符与数符相同　　B. 阶符与数符相异

C. 数符与尾数最高有效数位相同　　D. 数符与尾数最高有效数位相异

16. 两个浮点数相加，一个数的阶码值为 7，另一个数的阶码值为 9，则需要将阶码值较小的浮点数的小数点________。

A. 左移 1 位　　B. 右移 1 位　　C. 左移 2 位　　D. 右移 2 位

17. 4 片 74181 ALU 和 1 片 74182 CLA 相配合，具有________传递功能。

A. 串行进位　　B. 组内并行进位，组间串行进位

C. 组内串行进位，组间并行进位　　D. 组内、组间均为并行进位

18. 运算器虽由许多部件组成，但核心部件是________。

A. 算术逻辑运算单元　　B. 多路开关

C. 数据总线　　D. 累加寄存器

19. 下列叙述中，错误的是________。

A. 运算器中通常都有一个状态标志寄存器,为计算机提供判断条件,以实现程序转移

B. 补码乘法器中,被乘数和乘数的符号都不参加运算

C. 并行加法器中高位的进位依赖于低位

D. 在小数除法中,为了避免溢出,要求被除数的绝对值小于除数的绝对值

20. 计算机中的累加器________。

A. 没有加法器功能,也没有寄存器功能

B. 没有加法器功能,有寄存器功能

C. 有加法器功能,没有寄存器功能

D. 有加法器功能,也有寄存器功能

三、判断题

1. 进位信号串行传递的加法器称为串行加法器。 ( )
2. 进位产生函数为 $P_i=A_i \oplus B_i$。 ( )
3. 运算器中设置了加法器后就没有必要再设置减法器。 ( )
4. 浮点数对阶的原则是大阶向小阶看齐。 ( )
5. 运算器不仅可以完成数据信息的算逻运算,还可以作为数据信息的传送通路。 ( )
6. 80387 被称为协处理器,本身不能单独使用。 ( )

四、简答题

1. 简述浮点运算中溢出的处理问题。
2. 试述先行进位解决的问题及基本思想。

五、分析题

1. 某加法器采用组内并行、组间并行的进位链,4 位一组,写出进位信号 C_6 的逻辑表达式。

2. 写出一位 2421 码加法器的校正函数。

六、设计题

1. 已知某数的二进制原码表示为$[A]_原=A_nA_{n-1}\cdots A_1A_0$,其中 A_n 为符号位。请设计求$[A]_补$的逻辑电路。

2. 试用 74181 和门电路实现一位余 3 码加法器。

3. 设计一个 1 位 ALU,完成一位加法、AND、OR 和 NOT 操作。输入为 A、B,输出为 Z。当加法运算时,有进位输出 Carry Out;当执行 AND、OR 和 NOT 操作时,Carry Out 为 0。在图 4-16 上通过连线完成上述设计。(注:不能添加任何其他部件)

4.4.2 同步测试习题解答

一、填空题

1. 进位信号的产生和传递问题。
2. D0H,10H。假设题中 A 和 B 都用补码表示。
3. 11101010,11111010。
4. 左移 1 位,减 1。
5. 算术,逻辑。

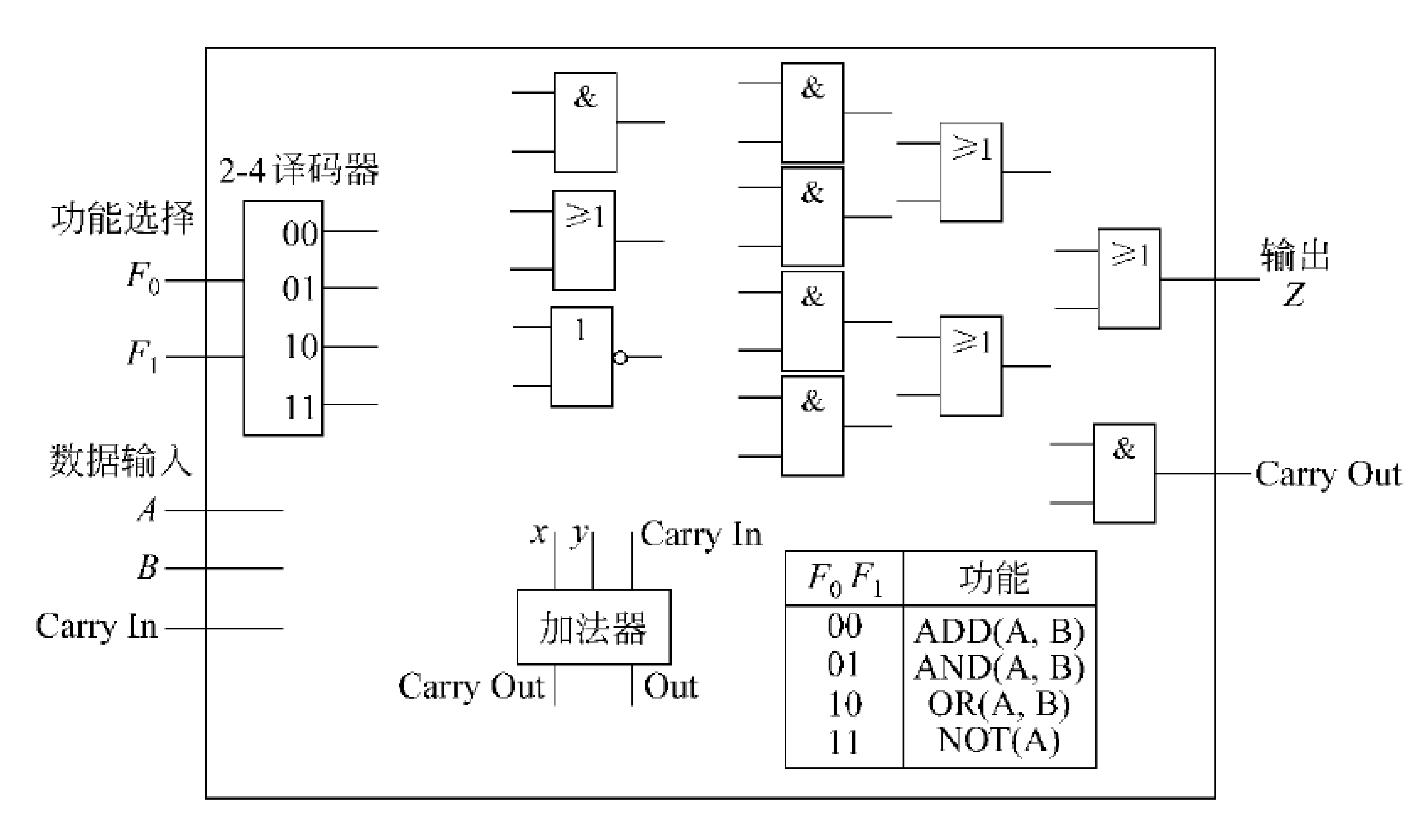


图 4-16　1 位 ALU 所需器件

二、选择题

1. C。本题中 4 个选项均会对加法器的速度产生影响，但只有进位传递延迟对并行加法器的影响最为关键。

2. A。进位产生函数 $G_i = A_i \cdot B_i$。

3. C。根据补码加减运算规则。

4. D。采用 1 位符号位判断溢出的方法有两个，其中之一与进位位有关系，判断条件是 $C_s \oplus C_1$。

5. B。正溢出时双符号位为 01，负溢出时双符号位为 10。

6. D。当运算结果超出机器所能表示的范围就发生溢出。

7. C。定点运算结果一旦发生溢出只有产生中断向 CPU 报错。

8. A。带符号补码的扩展，是用符号位填充高位。

9. B。十进制数 −121 的 8 位二进制补码表示为 10000111，扩展成 16 位二进制补码，符号扩展，表示为 1111111110000111。

10. A。C6H=11000110B，$[X]_补 = \left[\frac{X}{2}\right]_补 \times 2$，11000110 左移 1 位，变成 10001100=8CH。

11. C。注意，题干中是小数点左移或右移，而不是数左移或右移。

12. C。Booth 乘法需要做 $n+1$ 次累加，n 次移位。

13. C。由于被除数、除数取的都是绝对值，那么最终的余数当然应是正数。如果最后一步余数为负，则应将该余数加上除数，将余数恢复为正数，称为恢复余数。

14. B。乘法时 B 寄存器的初值是被乘数，除法时 B 寄存器的初值是除数。

15. D。补码表示的浮点数判断规格化的条件是数符与尾数最高位相异。

16. C。尾数右移相当于小数点左移，注意，题干中问的是小数点移位的次数，而不是尾数移位的次数。

17. D。两级分组并行进位，组内并行，组间也并行。

18. A。运算器的核心部件是算术逻辑运算单元(ALU)。

19. B。补码运算时,被操作数和操作数的符号位参加运算,结果的符号位自动形成。
20. B。累加器又称为累加寄存器,它实质上是寄存器,没有加法器的功能。

三、判断题

1. ×。串行加法器并没有进位传递问题。注意,不要将串行进位方式和串行加法器概念混淆。
2. ×。进位产生函数为 $G_i=A_iB_i$。
3. √。
4. ×。浮点数对阶的原则是小阶向大阶看齐。
5. √。
6. √。

四、简答题

1. 溢出就是超出了机器数所能表示的数据范围。浮点数范围是由阶码决定的,当运算以后的阶码大于所能表示的最大阶码值时,属于溢出(依尾数的正、负数决定是正溢出还是负溢出),当运算以后的阶码小于所能表示的最小阶码值时,计算机按机器零处理。

2. 先行进位解决的问题是进位的传递速度问题。其基本思想是:让每一位的进位与其低一位的进位无关,仅与两个参加操作的数以及最低位的进位有关。由于每位的操作数是同时给出的,各进位信号几乎可以同时产生,和数也随之产生,所以先行进位可以提高进位的传递速度,从而提高加法器的运算速度。

五、分析题

1. 最低一组的进位输出 $C_4=G_1^*+P_1^*C_0$。

其中:$G_1^*=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1$

$P_1^*=P_4P_3P_2P_1$

$C_5=G_5+P_5C_4$

所以,$C_6=G_6+P_6C_5=G_6+P_6G_5+P_6P_5C_4$。

2. 两个十进制数的2421码相加时,先按二进制数求和,然后校正。2421码的校正关系如表4-4所示。

表4-4　2421码的校正关系

十进制数	2421码					校正前的二进制数					校正关系
	C_4	S_4	S_3	S_2	S_1	C_4'	S_4'	S_3'	S_2'	S_1'	
0	0	0	0	0	0	0	0	0	0	0	不校正
1	0	0	0	0	1	0	0	0	0	1	
2	0	0	0	1	0	0	0	0	1	0	
3	0	0	0	1	1	0	0	0	1	1	
4	0	0	1	0	0	0	0	1	0	0	
5	0	1	0	1	1	0	0	1	0	1	+6校正
6	0	1	1	0	0	0	0	1	1	0	
7	0	1	1	0	1	0	0	1	1	1	
8	0	1	1	1	0	0	1	0	0	0	
9	0	1	1	1	1	0	1	0	0	1	

续表

十进制数	2421码					校正前的二进制数					校正关系
	C_4	S_4	S_3	S_2	S_1	C'_4	S'_4	S'_3	S'_2	S'_1	
10	1	0	0	0	0	1	0	1	1	0	−6校正
11	1	0	0	0	1	1	0	1	1	1	
12	1	0	0	1	0	1	1	0	0	0	
13	1	0	0	1	1	1	1	0	0	1	
14	1	0	1	0	0	1	1	0	1	0	
15	1	1	0	1	1	1	1	0	1	1	不校正
16	1	1	1	0	0	1	1	1	0	0	
17	1	1	1	0	1	1	1	1	0	1	
18	1	1	1	1	0	1	1	1	1	0	
19	1	1	1	1	1	1	1	1	1	1	

根据校正关系，很容易得到校正函数：

$$+6\text{校正}=\overline{C'_4}(\overline{S'_4}S'_3S'_2+\overline{S'_4}S'_3S'_1+S'_4\overline{S'_3}\,\overline{S'_2})$$

$$-6\text{校正}=C'_4(\overline{S'_4}S'_3S'_2+S'_4\,\overline{S'_3}\,\overline{S'_2}+S'_4\overline{S'_3}\,\overline{S'_1})$$

六、设计题

1. 原码求补码的方法是：$A_n=0$，$[A]_原=[A]_补$；$A_n=1$，从数的最低位开始，由右向左，直到找出第一个“1”。若 $A_i=1$，A_i 右侧的每一位，包括 A_i 自身，都保持不变，而 A_i 左侧的每一位都变反。为此，采用按位扫描技术来执行所需要的操作，逻辑电路如图 4-17 所示。

图 4-17 中 E 为使能控制端。当 $E=0(A_n=0)$，输出与输入相等，即正数的原码与正数的补码相等；当 $E=1(A_n=1)$，从 $A_i=1$ 处向左，各位开始变反。

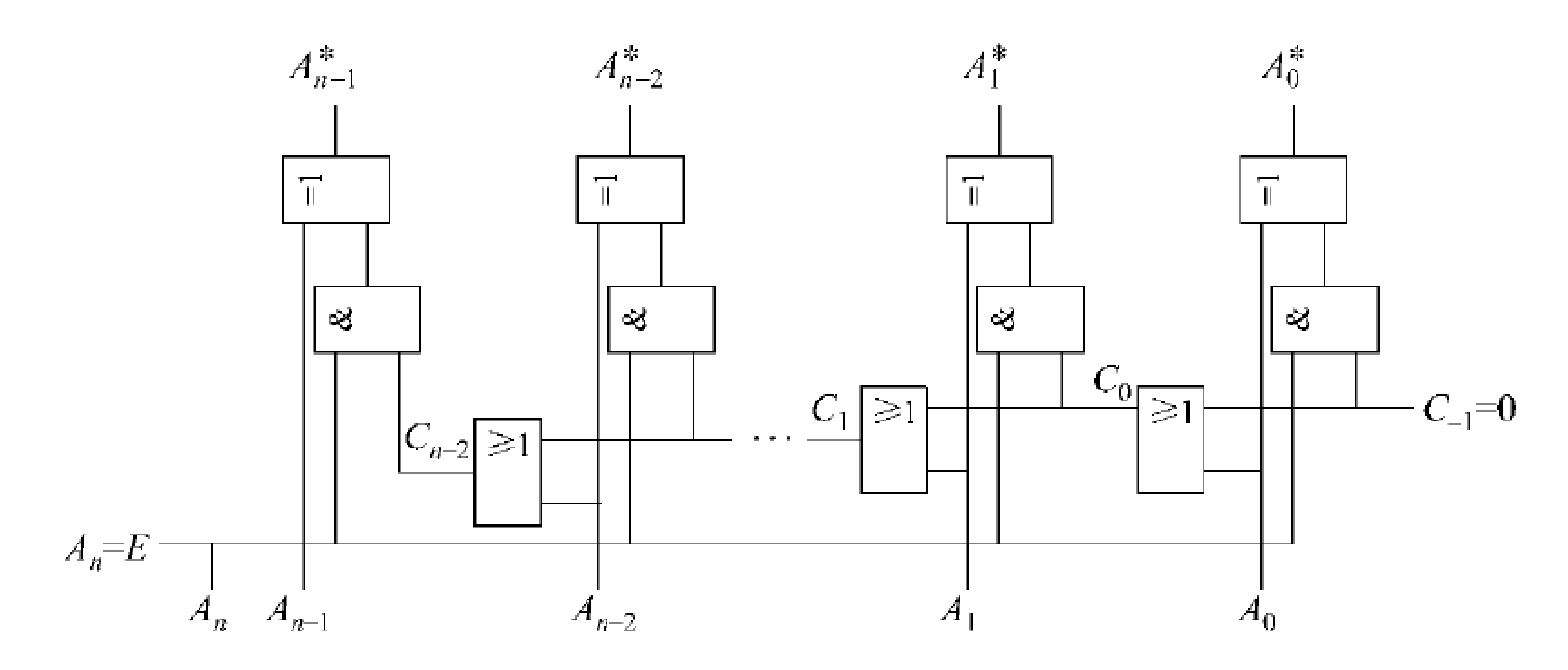


图 4-17　二进制求补逻辑电路

2. 根据余 3 码的校正函数，用 74181 和一个与门电路可以实现 1 位余 3 码加法器，其逻辑框图如图 4-18 所示。有进位，+3(+0011)校正；无进位，−3(+1101)校正。

3. 能完成加法、AND、OR 和 NOT 功能的 1 位 ALU 如图 4-19 所示。

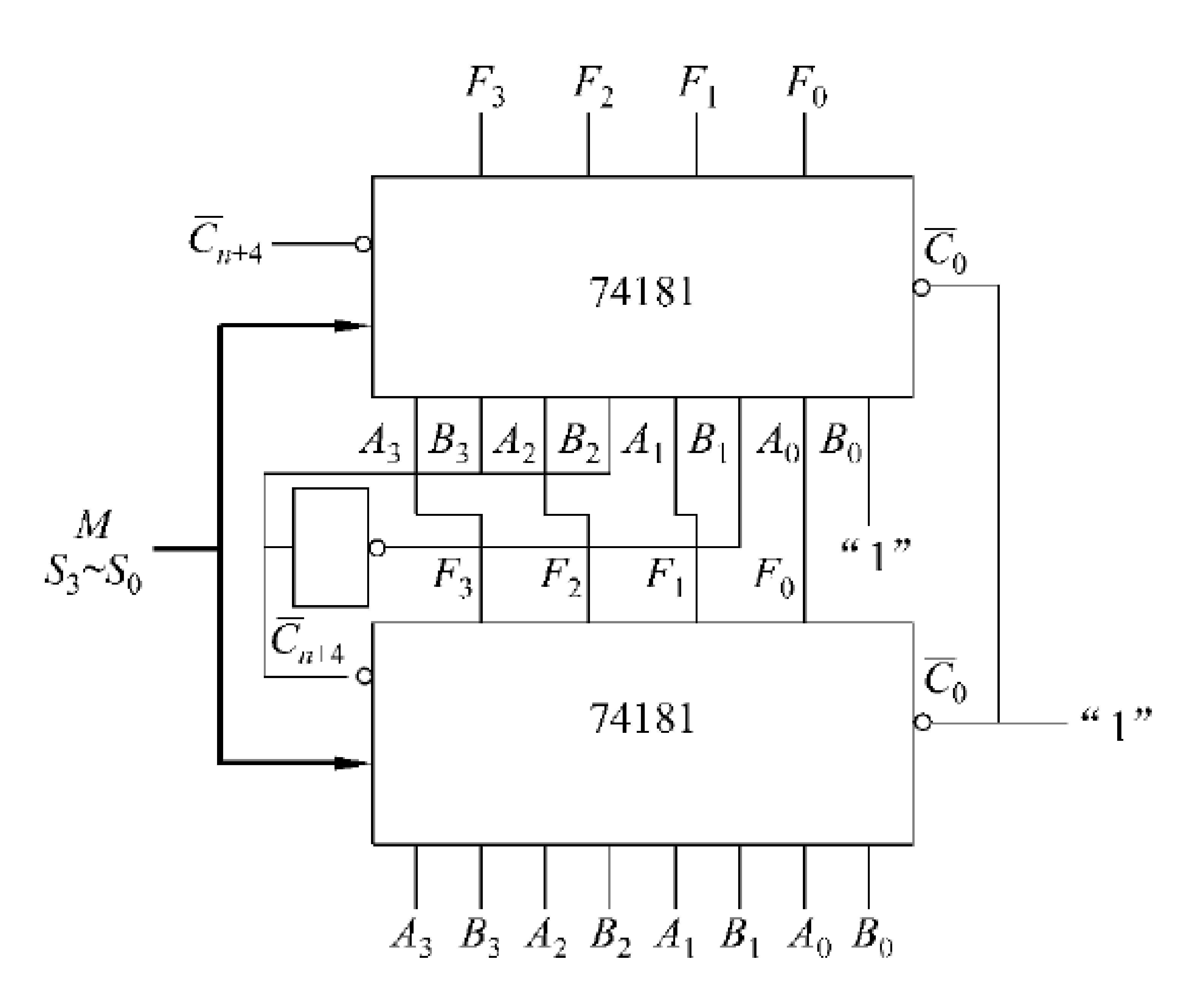


图 4-18　用 74181 实现余 3 码加法器

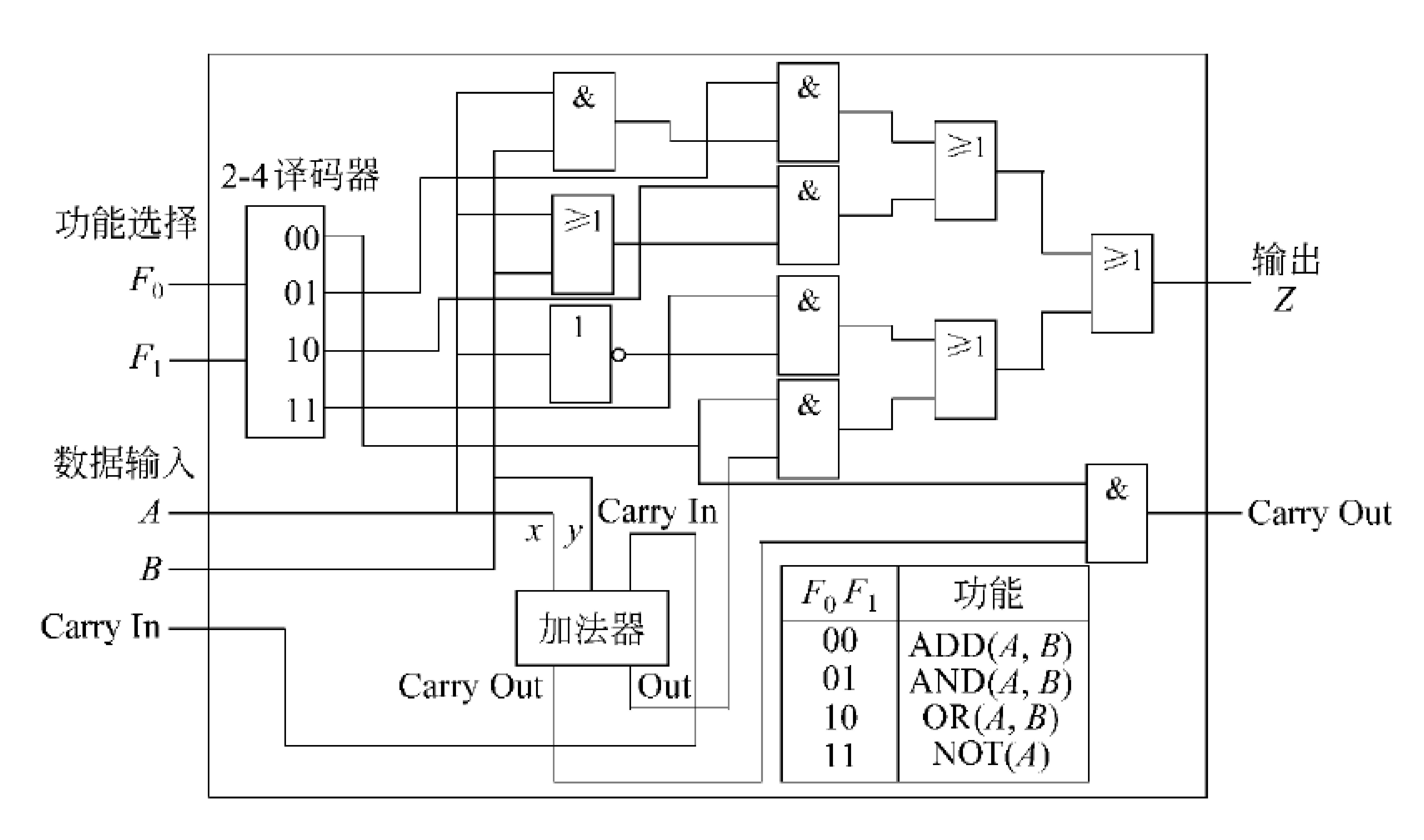


图 4-19　1 位 ALU 连接图

第 5 章 存储系统和结构

5.1 基本内容摘要

- 存储系统的组成
  - ◆ 存储器分类
  - ◆ 存储系统层次结构

    Cache—主存存储层次；

    主存—辅存存储层次。
- 主存储器的组织
  - ◆ 主存储器的基本结构

    主存通常由存储体、地址译码驱动电路、I/O 和读写电路组成。
  - ◆ 主存储器的存储单元

    大端方案；

    小端方案。
  - ◆ 主存储器的主要技术指标

    存储容量；

    存取速度。
  - ◆ 数据在主存中的存放

    不浪费存储器资源的存放方法；

    从一个存储字的起始位置开始存放方法；

    边界对齐的存放方法。
- 半导体随机存储器和只读存储器
  - ◆ RAM 记忆单元电路
  - ◆ 动态 RAM 的刷新

    集中刷新方式；

    分散刷新方式；

    异步刷新方式。
  - ◆ RAM 芯片分析
  - ◆ 半导体只读存储器(ROM)

    掩膜式 ROM(MROM)；

一次可编程 ROM(PROM);
可擦除可编程 ROM(EPROM);
闪速存储器。

- 主存储器的连接与控制
  - 主存容量的扩展
    位扩展法、字扩展法、字和位同时扩展法。
  - 存储芯片的地址分配和片选
  - 主存储器和 CPU 的连接
  - PC 系列微机的存储器接口
- 提高主存读写速度的技术
  - 主存与 CPU 速度的匹配
- 多体交叉存储技术
  - 并行访问存储器
  - 交叉访问存储器
- 高速缓冲存储器
  - 高速缓存工作原理
  - Cache 的读写操作
  - 地址映像
  - 替换算法
  - 更新策略
- 虚拟存储器
  - 虚拟存储器的基本概念
  - 页式虚拟存储器
  - 快表与慢表

5.2　重点难点梳理

1. 主存储器的编址方式

早期计算机的主存储器(简称主存)均按字编址,CPU 在访问主存时,给出一个地址码,就能从主存中读出一个字长的信息或者将一个字长的信息写入主存中。这样的存储器称为按字编址的存储器。

随着计算机技术的飞速发展,存储器的字长不断增长。例如,存储器字长为 32 位,如果仍然采用按字编址方式,则每访问一次存储器只能读出和写入 32 位信息。32 位包含 4 个字节,如果能将访问存储器的地址指向一个字中的每个字节,就能大大增强访问存储器的灵活性,使得访问一次存储器可读出或写入任何一个字节的信息,也可读出或写入整个字的信息,这就要求主存按字节编址。

某系统中主存容量为 16K×32,如果采用按字编址方式,地址码长度应为 14 位,如果采用按字节编址,则地址码的长度应为 16 位,如图 5-1 所示。

图 5-1(a)和图 5-1(b) 所示的存储器容量相同,都是 16K×32,区别仅在于图 5-1(a)是

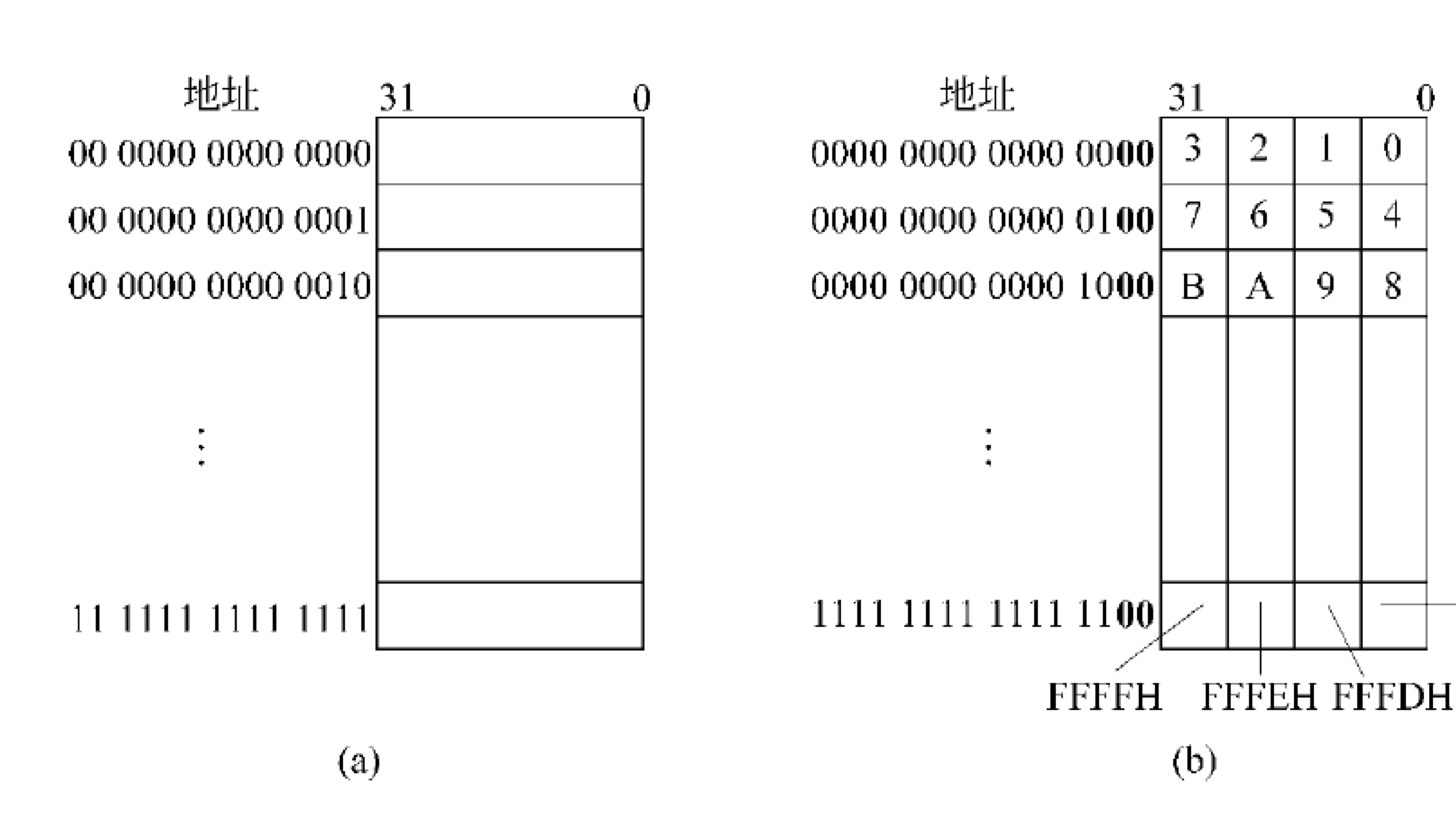


图 5-1　按字编址与按字节编址的区别

按字编址，每个地址指向一个存储字；而图 5-1(b)是按字节编址，每个地址指向一个字节。显然，由于一个存储字中包含 4 个字节，所以，按字节编址时，其地址码的长度要比按字编址的地址码长 2 位，这 2 位为 00、01、10、11，分别指向一个存储字中的 4 个字节。图 5-1(a)中的字地址码是连续的，而图 5-1(b)中的字地址码是不连续的，这是由于它所指向的是该字中最低端的那个字节。

采用按字节编址的存储器可根据给定的字节地址访问存储器中的任何一个字节，也可以根据给定的字地址访问一个存储字。

2. 操作数的存储方式

一个多字节的数据在按字节编址的主存中通常由两种排序方案——大端次序和小端次序。大端次序方案将最高有效字节存储在最小地址位置，小端次序方案将最低有效字节存储在最小地址位置。图 5-2 是 32 位的十六进制数 12345678 在存储器中的存储方式示意图。

Intel 80x86 是采用小端次序方案的机器，IBM 370、Motorola 680x0 和大多数 RISC 机器则采用大端次序方案。Power PC 是一个既支持大端方案又支持小端方案的机器。

地址	值
4	12
5	34
6	56
7	78

大端次序

地址	值
4	78
5	56
6	34
7	12

小端次序

图 5-2　多字节数据的两种存储形式

3. 存取时间和存取周期

存取时间(T_a)，又称为访问时间或读写时间，它是指从启动一次存储器操作到完成该操作所经历的时间。例如，读出时间是指从 CPU 向存储器发出有效地址和读命令开始，直到将被选单元的内容读出为止所用的时间；写入时间是指从 CPU 向存储器发出有效地址和写命令开始，直到信息写入被选中单元为止所用的时间。显然 T_a 越小，存取速度越快。

存取周期(T_m)，又可称作读写周期、访存周期，是指存储器进行一次完整的读写操作所需的全部时间，即连续两次访问存储器操作之间所需要的最短时间。显然，一般情况下，$T_m > T_a$。这是因为对于任何一种存储器，在读写操作之后，总要有一段恢复内部状态的复原时间。对于破坏性读出的存储器，存取周期往往比存取时间要大得多，甚至可以达到 $T_m = 2T_a$，这是因为存储器中的信息读出后需要马上进行重写(再生)。

与存取周期密切相关的指标是主存带宽，它又称为数据传输率，表示每秒从主存进出信

息的最大数量,单位为字节每秒或位每秒。

4. 边界对齐的数据存放方式

在数据对齐存储方式下,要求一个数据字占据完整的一个存储字的位置,而不是分成两部分各占据每个存储字位置的一部分。例如,一个32位的数据字放在32位宽度的主存储器中,若字地址为n,则在对齐方式下数据实际占据的是字节地址为n、$n+1$、$n+2$和$n+3$的存储单元,这个数据可以一次读取或者写入。如果这个数据字不按对齐方式存储,假设数据实际占据的是字节地址为$n-1$、n、$n+1$和$n+2$存储单元,这样的数据在32位的存储器中需要分两次读取或者写入。在有些计算机中,规定数据的存储必须按边界对齐的方式进行。图5-3(a)是一个边界不对齐的例子,图5-3(b)是一个边界对齐的例子。

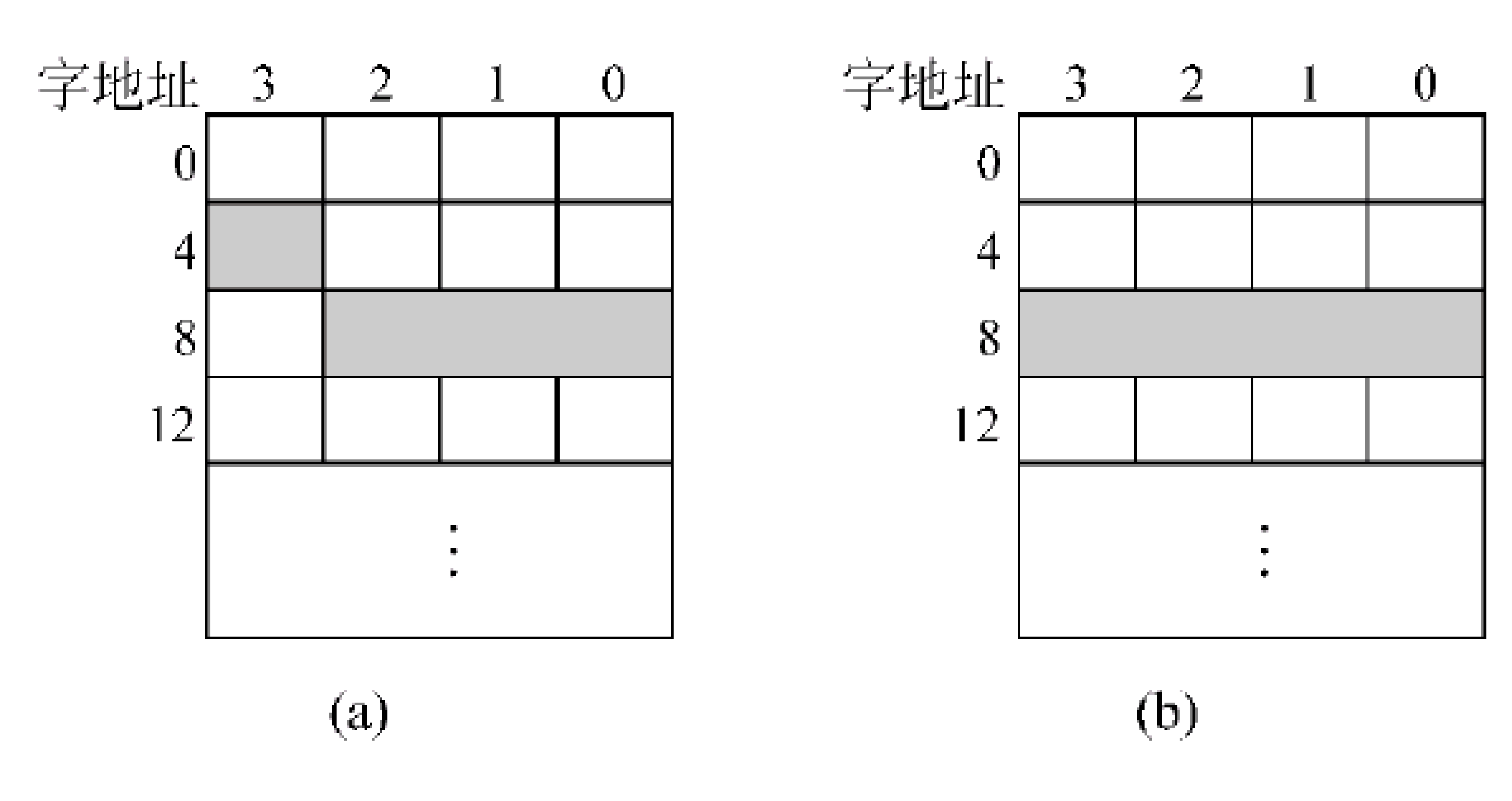


图5-3　32位的数据字在32位宽度主存储器中的存放

假设,某机存储字长为64位(8字节),读写的数据有4种不同长度,它们分别是字节(8位)、半字(16位)、单字(32位)和双字(64位)。

边界对齐的数据存放方式对数据的存放位置有下列要求:

字节数据的地址为×…××××(任意)

半字数据的起始地址为×…×××0(2的整倍数)

单字数据的起始地址为×…×× 0 0(4的整倍数)

双字数据的起始地址为×…× 0 0 0(8的整倍数)

5. 动态RAM的刷新

为了维持动态RAM记忆单元的存储信息,通常每隔2ms就必须对存储体中所有记忆单元的栅极电容补充一次电荷,即使许多记忆单元长期未被访问也是如此,这个过程就是刷新。

刷新和重写(再生)是两个完全不同的概念,切不可混淆。刷新仅针对动态RAM,刷新是定时进行的,即使动态RAM的许多记忆单元长期未被访问,若不及时补充电荷,信息也会丢失。重写则仅针对破坏性读出的存储器(如磁芯、单管动态RAM等),重写是随机进行的,某个存储单元只有在其被读取之后才需要重写。刷新通常是以存储体矩阵中的一行为单位进行的,而重写一般是以存储单元为单位进行的。

常见的刷新方式有集中式、分散式和异步式3种。

集中刷新方式的优点是读写操作时不受刷新工作的影响,因此系统的存取速度比较高。主要缺点是在集中刷新期间必须停止读写,这一段时间称为“死区”,而且存储容量越大,死区就越长。

分散刷新方式没有死区,但是它有两个明显的缺点,第一是加长了系统的存取周期,降

低了整机的速度;第二是刷新过于频繁,尤其是当存储容量比较小的情况下,没有充分利用所允许的最大刷新间隔(2ms)。

异步刷新方式是前述两种方式的结合,它充分利用了最大刷新间隔时间,把刷新操作平均分配到整个最大刷新间隔时间内进行。这种方式虽然也有死区,但比集中刷新方式的死区小得多,死区的长度仅等于一个存储周期。

6. DRAM 的刷新中要注意的几个问题

(1) 无论是由刷新控制逻辑产生地址循环码逐行循环地刷新,还是芯片内部自动地刷新,都不依赖于外部的访问,刷新对 CPU 是透明的。

(2) 刷新通常是一行一行地进行的,每一行中各记忆单元同时被刷新,故刷新操作时仅需要行地址,不需要列地址。

(3) 刷新操作类似于读出操作,但又有所不同。因为刷新操作仅是给栅极电容补充电荷,不需要信息输出。另外,刷新时不需要加片选信号,即整个存储器中的所有芯片同时被刷新。

(4) 因为所有芯片同时被刷新,所以在考虑刷新问题时,应当从单个芯片的存储容量着手,而不是从整个存储器的容量着手。

7. RAM 芯片结构

RAM 芯片通过地址线、数据线和控制线与外部连接。地址线是单向输入的,其数目与芯片容量有关。如容量为 1024×4 时,地址线有 10 根;容量为 64K×1 时,地址线有 16 根。数据线是双向的,既可输入,也可输出,其数目与数据位数有关。如 1024×4 的芯片,数据线有 4 根;64K×1 的芯片,数据线只有 1 根。控制线主要有读写控制线和片选线两种,读写控制线用来控制芯片是进行读操作还是写操作的,片选线用来决定该芯片是否被选中。

由于 DRAM 芯片集成度高,容量大,为了减少芯片引脚数量,DRAM 芯片把地址线分成相等的两部分,分两次从相同的引脚送入,两次输入的地址分别称为行地址和列地址。

RAM 芯片中的地址译码电路能把地址线送来的地址信号翻译成对应存储单元的选择信号。地址译码方式主要有下面两种。

单译码方式:又称字选法,它所对应的存储芯片结构是字结构的。对应任何一个地址码只有一条选择线(字线)有效,连接在这条字线上的所有记忆单元同时被选中。

双译码方式:又称重合法,它所对应的存储芯片结构可以是位结构的,也可以是字段结构的。通常是把 K 位地址码分成接近相等的两段,一段用于水平方向作 X 地址线,供 X 地址译码器译码;一段用于垂直方向作 Y 地址线,供 Y 地址译码器译码。对应任何一个地址码只有一条 X 选择线和一条 Y 选择线有效,其交叉处的一个记忆单元被选中。

小容量的 RAM 宜采用单译码方式,它对选择线的负载能力要求不高,但选择线数量多;大容量的 RAM 宜采用双译码方式,它的选择线比较少,但每条选择线的负载较重。

8. 用若干芯片构成主存储器

主存储器是整个存储系统的核心,通常分为随机存储器(RAM)和只读存储器(ROM)两大部分。RAM 和 ROM 在主存中是统一编址的。

存储芯片的容量是有限的,主存储器往往要由一定数量的芯片构成。根据主存所要求的容量和选定的存储芯片的容量,就可以计算出总的芯片数,即

$$总片数 = \frac{总容量}{容量/片}$$

1）位扩展

当主存的字数与单个存储芯片的字数相同而位数不同时，可采用位扩展方式来组织多个芯片构成主存。

位扩展仅在位数方向扩展(加大字长)，位扩展的连接方式是将各存储芯片的地址线、片选线和读写线相应地并联起来，而将各芯片的数据线单独列出。

2）字扩展

当主存的字长与单个存储芯片的字长相同而字数不同时，可采用字扩展方式来组织多个芯片构成主存。

字扩展仅在字数方向扩展，而位数不变。字扩展将芯片的地址线、数据线、读写线并联，由片选信号来区分各个芯片。字扩展时有一个地址分配问题，地址线的高位部分通过译码器产生若干个片选信号 CS_i，分别选中若干个芯片中的一个。

3）字和位同时扩展

实际中更多出现的是存储芯片的字数和字长均不能满足主存总容量要求的情况，这时要采用字和位同时扩展方式来构成主存。

9. CPU 的访存地址

CPU 访问主存时需要给出地址码，其长度取决于 CPU 可直接访问的最大存储空间，一般要将其地址码分成片内地址和选片地址两部分。

片内地址由低位的地址码构成，其长度取决于所选存储芯片的字数，例如，芯片容量为 8K×4 和 8K×1，它们的片内地址相同，均为 13 位($2^{13}=8K$)，片内地址用于从选中的芯片中选择出相应的存储单元，以进行数据的存取，故又称为字选。

选片地址由高位的地址码构成，用于选择存储芯片，故又称为片选地址。大多数情况下由选片地址通过译码后产生存储芯片的片选信号($\overline{CS}$)。

10. 选片地址的全译码方式

所谓全译码方式是指所有的选片地址全部参加译码，有两种情况必须采用全译码方式。

(1) 如果实际使用的存储空间与 CPU 可访问的最大存储空间相同。

例如，CPU 给出的访存地址长 16 位($A_{15}\sim A_0$)，即可访问的最大存储空间为 64KB，选用的存储芯片容量为 16K×4，共 8 片，构成 4 个小组，这时片内地址为 14 位，选片地址为 2 位，2 位选片地址必须全部参加译码才能产生 4 个选片信号，分别用作 4 个小组的选片信号。

(2) 如果实际使用的存储空间小于 CPU 可访问的最大存储空间，而对实际空间的地址分配有严格的要求。

例如，CPU 给出的访存地址长 16 位($A_{15}\sim A_0$)，即可访问的最大存储空间为 64KB，而系统中实际使用的存储空间只有 8KB，且选用存储容量为 4K×2 的芯片共 8 片，并要求其地址范围必须在 4000H～5FFFH 范围内，其地址译码方式如图 5-4 所示。

从图 5-4 可以看出，其地址分配如表 5-1 所示。

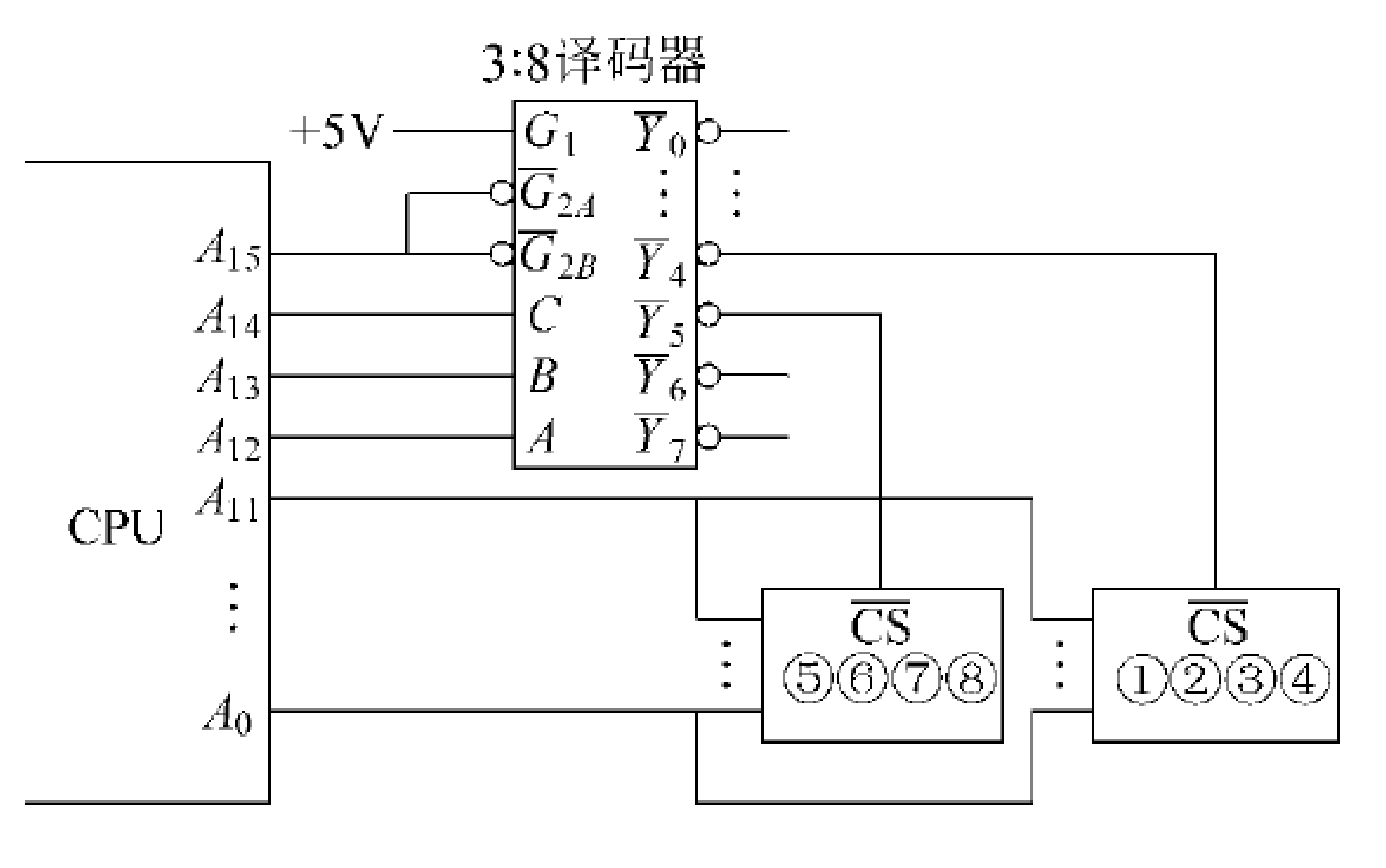


图 5-4　地址码采用全译码方式

表 5-1　全译码方式的地址分配

所选芯片号	选片地址				片内地址					译码输出	地址分配
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	…	A_0		
	0	0	0	0	0	0	0	…	0	$\overline{Y}_0=0$	0000H～0FFFH
	0	0	0	0	1	1	1	…	1		
	0	0	0	1	0	0	0	…	0	$\overline{Y}_1=0$	1000H～1FFFH
	0	0	0	1	1	1	1	…	1		
	0	0	1	0	0	0	0	…	0	$\overline{Y}_2=0$	2000H～2FFFH
	0	0	1	0	1	1	1	…	1		
	0	0	1	1	0	0	0	…	0	$\overline{Y}_3=0$	3000H～3FFFH
	0	0	1	1	1	1	1	…	1		
①②③④	0	1	0	0	0	0	0	…	0	$\overline{Y}_4=0$	4000H～4FFFH
	0	1	0	0	1	1	1	…	1		
⑤⑥⑦⑧	0	1	0	1	0	0	0	…	0	$\overline{Y}_5=0$	5000H～5FFFH
	0	1	0	1	1	1	1	…	1		
	0	1	1	0	0	0	0	…	0	$\overline{Y}_6=0$	6000H～6FFFH
	0	1	1	0	1	1	1	…	1		
	0	1	1	1	0	0	0	…	0	$\overline{Y}_7=0$	7000H～7FFFH
	0	1	1	1	1	1	1	…	1		

从表 5-1 中可以看出，按照这种译码方式，当前使用的存储空间的地址范围被严格地定义在 4000H～5FFFH 范围内，最大可扩充到 32KB 的存储空间。如果要将主存储器容量扩充到 64KB，只需将译码电路稍作修改即可。

全译码方式的共同特点是所使用的存储芯片的地址范围是唯一的。

11. 选片地址的部分译码方式

当实际使用的存储空间比 CPU 可访问的最大存储空间小，而且对其地址分配没有严格要求的情况下可采用部分译码方式。

例如，CPU 可提供的地址为 16 位(A_{15}～A_0)，而实际使用的存储空间为 16KB，拟采用 4K×4 的存储芯片共 8 片，则可采用的部分译码方式如图 5-5 所示。

由于采用部分译码方式，使得各组芯片的地址访问不再是唯一的。每组芯片的地址分配分别为：

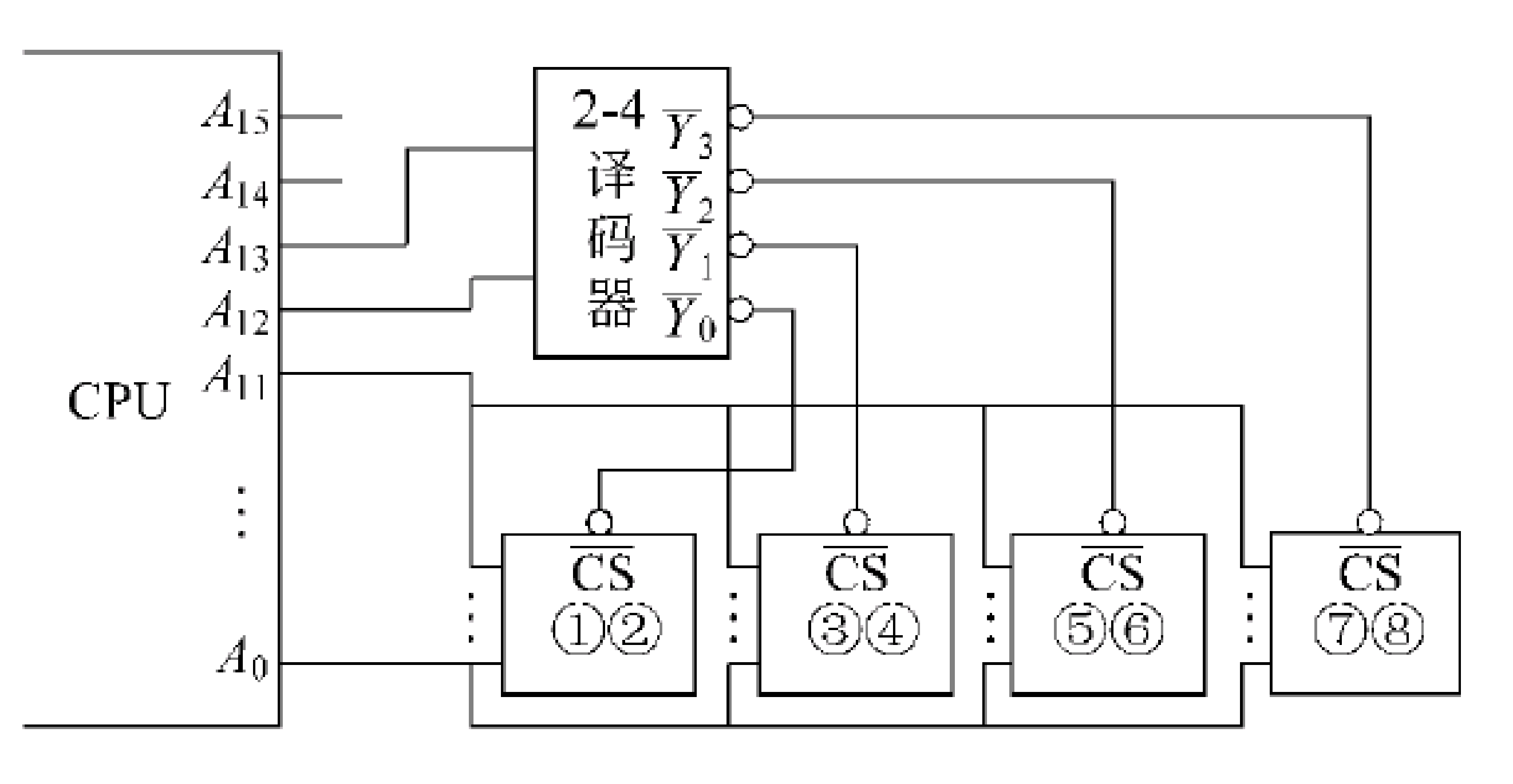


图 5-5　地址码采用部分译码方式

第 1 组：0000H～0FFFH,4000H～4FFFH,8000H～8FFFH,C000H～CFFFH

第 2 组：1000H～1FFFH,5000H～5FFFH,9000H～9FFFH,D000H～DFFFH

第 3 组：2000H～2FFFH,6000H～6FFFH,A000H～AFFFH,E000H～EFFFH

第 4 组：3000H～3FFFH,7000H～7FFFH,B000H～BFFFH,F000H～FFFFH

可以看出,采用部分译码方式的结果,使得各组芯片出现了重叠的地址范围,其地址重叠区的个数取决于没有参加译码的地址码的位数,由于有 2 位地址码(A_{15}、A_{14})没有参加译码,所以每组芯片都出现 4 个地址重叠区。

12. CPU 对主存的基本操作

CPU 对主存进行读写操作时,首先 CPU 在地址总线上给出地址信号,然后发出相应的读或写命令,并在数据总线上交换信息。读写的基本操作如下。

1) 读

读操作是指从 CPU 送来的地址所指定的存储单元中取出信息,再送给 CPU,其操作过程是：

(1) 地址→MAR→AB	CPU 将地址信号送至地址总线；
(2) Read	CPU 发读命令；
(3) Wait for MFC	等待存储器工作完成信号；
(4) M (MAR)→DB→MDR	读出信息经数据总线送至 CPU。

2) 写

写操作是指将要写入的信息存入 CPU 所指定的存储单元中,其操作过程是：

(1) 地址→MAR→AB	CPU 将地址信号送至地址总线；
(2) 数据→MDR→DB	CPU 将要写入的数据送至数据总线；
(3) Write	CPU 发写命令；
(4) Wait for MFC	等待存储器工作完成信号。

13. 双口 RAM

双口 RAM 是指同一个存储器具有两组相互独立的读写控制电路,是一种高速工作的存储器。它有两个独立的端口,分别具有各自的地址线、数据线和控制线,可以对存储器中任何位置上的数据进行独立的存取操作。

双口 RAM 的核心部分是用于数据存储的存储器阵列,可为左、右两个端口所共用。当两个端口的地址不相同时,在两个端口上进行读写操作,一定不会发生冲突。当任一端口被选中驱动时,就可对整个存储器进行存取,每一个端口都有自己的片选控制和输出驱动

控制。

当两个端口同时存取存储器的同一存储单元时，就会因数据冲突造成数据存储或读取错误。两个端口对同一主存操作有4种情况：

（1）两个端口不同时对同一地址单元存取数据；

（2）两个端口同时对同一地址单元读出数据；

（3）两个端口同时对同一地址单元写入数据；

（4）两个端口同时对同一地址单元，一个写入数据，另一个读出数据。

在第（1）、第（2）种情况时，两个端口的存取不会出现错误，第（3）种情况会出现写入错误，第（4）种情况会出现读出错误。为避免第（3）、（4）种错误情况的出现，双口RAM设计有硬件 $\overline{BUSY}$ 功能输出，其工作原理如下：当左、右端口不对同一地址单元存取时，$\overline{BUSY_R}$=H，$\overline{BUSY_L}$=H，可正常存储。当左、右端口对同一地址单元存取时，有一个端口的 $\overline{BUSY}$=L，禁止数据的存取。此时，两个端口中，哪个存取请求信号出现在前，则其对应的 $\overline{BUSY}$=H，允许存取；哪个存取请求信号出现在后，则其对应的 $\overline{BUSY}$=L，禁止其写入数据。需要注意的是，两端口间的存取请求信号出现时间要相差在5ns以上，否则仲裁逻辑无法判定哪一个端口的存取请求信号在前；在无法判定哪个端口先出现存取请求信号时，控制线 $\overline{BUSY_L}$ 和 $\overline{BUSY_R}$ 只有一个为低电平，不会同时为低电平。这样，就能保证对应于 $\overline{BUSY}$=H 的端口能进行正常存取，对应于 $\overline{BUSY}$=L 的端口不存取，从而避免双端口存取出现错误。

14. 多模块存储器

多模块存储器的每个模块具有相同的容量和存取速度，各模块都有独立的地址寄存器、数据寄存器、地址译码、驱动电路和读写电路，它们既能并行工作，又能交叉工作。

多模块交叉存储器是线性编址的，地址在各模块中有两种安排方式，分别是高位交叉编址（顺序方式）和低位交叉编址（交叉方式）。

高位交叉编址的多模块存储器用地址码的高位区分存储模块，地址码的低位选择存储单元。低位交叉编址的多模块存储器用地址码的低位区分存储模块，地址码的高位选择存储单元。

15. 程序访问的局部性原理

程序的局部性有两个方面的含义：时间局部性和空间局部性。时间局部性是指如果一个存储单元被访问，则可能该单元会很快被再次访问。这是因为程序存在着循环。空间局部性是指如果一个存储单元被访问，则该单元邻近的单元也可能很快被访问。这是因为程序中大部分指令是顺序存储、顺序执行的，数据一般也是以向量、数组、树、表等形式簇聚地存储在一起的。

也就是说，最近的、未来要用的指令和数据大多局限于是正在使用的指令和数据，或是存放在与这些指令和数据位置上邻近的单元中。这样，就可以把目前常用或将要用到的信息预先放在存取速度最快的存储器 M_1 中，从而使CPU的访问速度大大提高。

CPU访存时的基本原则：由近到远，首先访问 M_1，若在 M_1 中找不到所要的数据，就要访问 M_2，将包含所需数据的块或页面调入 M_1。若在 M_2 中还找不到，就要访问 M_3，依次类推。

16. Cache 的基本工作原理

利用程序的局部性原理，把程序中正在使用的部分存放在一个高速的容量较小的

Cache中,使CPU的访存操作大多数针对Cache进行,从而使程序的执行速度大大提高。

Cache和主存都被分成若干个大小相等的块,每块由若干字节组成。由于Cache的容量远小于主存的容量,所以Cache中的块数要远少于主存中的块数,它保存的信息只是主存中最急需执行的若干块的副本。当CPU发出主存地址后,首先判断该存储字是否在Cache中,若命中,则直接访问Cache;若不命中,则访问主存并将该字所在的主存块装入Cache。

命中率H定义为CPU产生的逻辑地址能在Cache中访问到的概率。在一个程序执行期间,设N_1为访问Cache的命中次数,N_2为访问主存的次数。

$$H = \frac{N_1}{N_1 + N_2}$$

Cache-主存存储层次的等效访问时间T_A根据主存的启动时间有:

假设Cache访问和主存访问是同时启动的,$T_A = H \times T_{A_1} + (1-H) \times T_{A_2}$。

假设Cache不命中时才启动主存,$T_A = T_{A_1} + (1-H) \times T_{A_2}$。

存储层次的访问效率$e = \dfrac{T_{A_1}}{T_A}$。

17. Cache和主存之间的映射方式

由于Cache的容量小,因此Cache中的内容会经常地被新的主存块替换掉,它们之间就有一个地址映射问题。常见的地址映射的方法有3种:全相联映射、直接映射和组相联映射。

全相联映射就是让主存中任何一个块均可以映射装入到Cache中任何一个块的位置上。全相联映射方式比较灵活,Cache的块冲突概率最低、空间利用率最高,但是地址变换速度慢,而且成本高,实现起来比较困难。

直接映射是指主存中的每一个块只能被放置到Cache中唯一的一个指定位置,若这个位置已有内容,则产生块冲突,原来的块将无条件地被替换出去。直接映射方式是最简单的地址映射方式,成本低,易实现,地址变换速度快,但这种方式不够灵活,Cache的块冲突概率最高、空间利用率最低。

组相联映射实际上是全相联映射和直接映射的折中方案,所以其优点和缺点介于全相联和直接映射方式之间。当组数等于1(不再分组),组相联映射就变成全相联映射;当组数等于Cache中块的数目,组相联映射就变成直接映射。

通常将组内2块的组相联映射称为二路组相联,组内4块的组相联映射称为四路组相联。关于组相联映射方式的具体映射实现方案,目前在不同的教材中有两种不同的说法,以二路组相联为例,图5-6(a)所示的方案称为方案一,图5-6(b)所示的方案称为方案二。

例如,主存的第9块,按方案一,将映射到Cache的第1组中,放在Cache的第2块或第3块的位置上;而按方案二,将映射到Cache的第0组中,放在Cache的第0块或第1块的位置上。比较这两个方案可以发现,两者的区别在于主存是否要按照Cache的大小再分区。这两种方案对应的主存地址是有区别的,如图5-7所示。

图5-7(a)是方案一对应的主存地址,分为3个字段。图5-7(b)是方案二对应的主存地址,分为4个字段,其中组内块号字段指出组相联中的一个Cache组(行)中块的数量,也就是组相联中的"路数"。相比之下,方案一的实现比较简单。组相联映射的关系可以定义为:

$$J = I \bmod Q$$

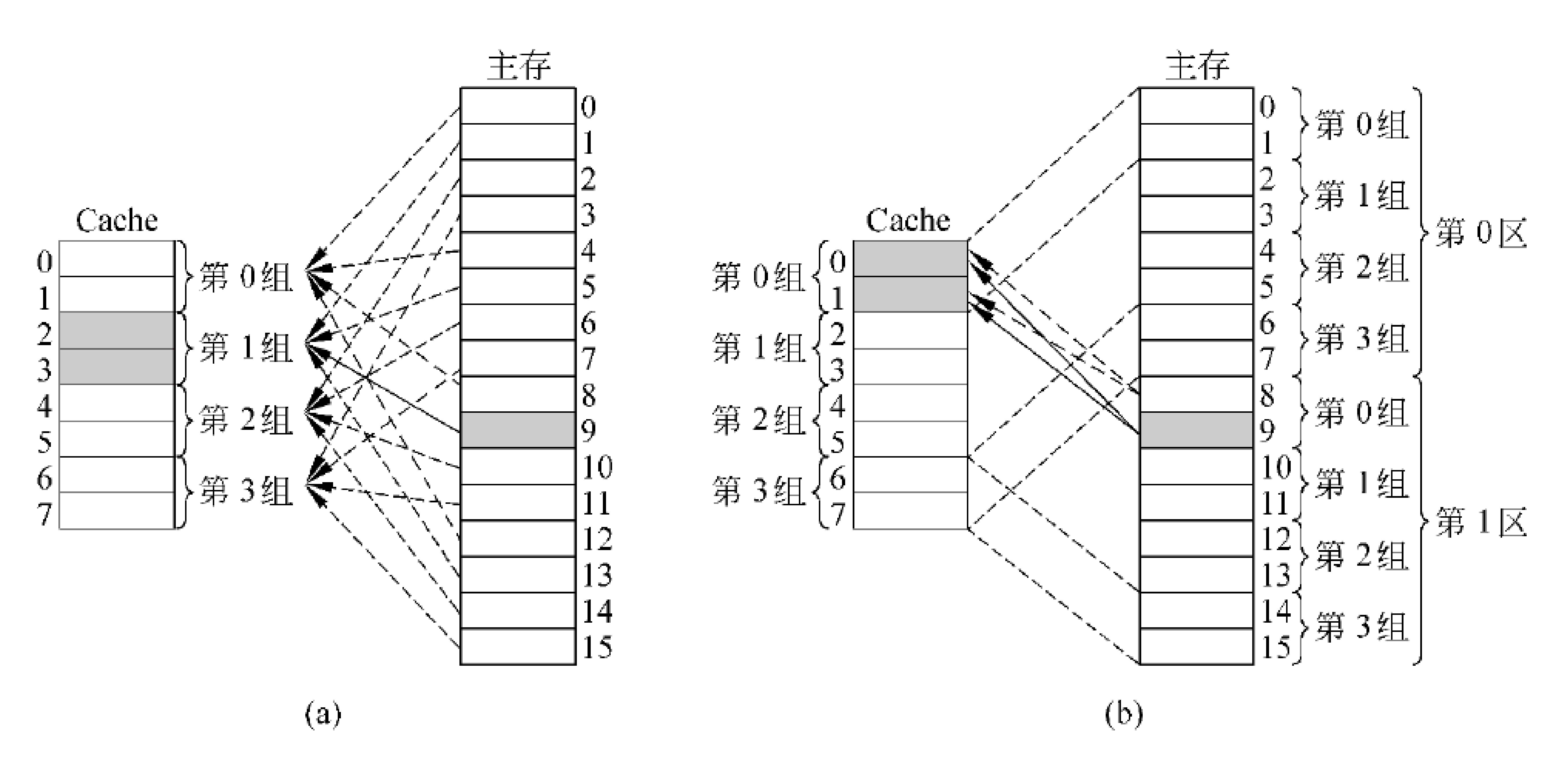


图 5-6　二路组相联映射方式的具体映射实现方案

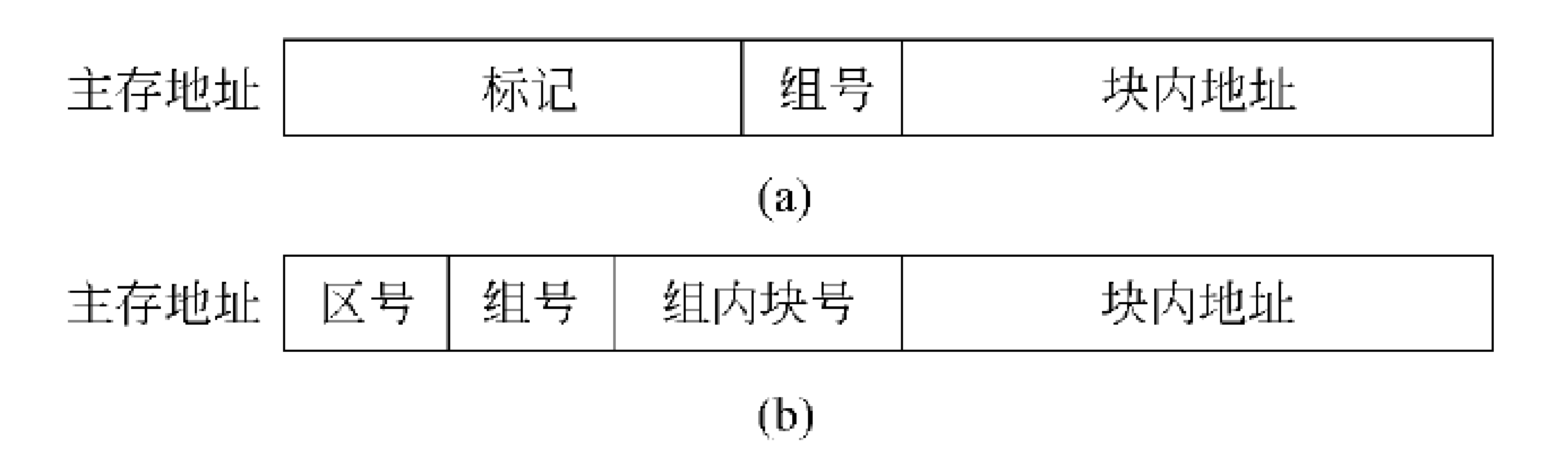


图 5-7　组相联映射方式的主存地址

式中，J 为 Cache 的组号；I 为主存的块号；Q 为 Cache 的组数。

18. 页式虚拟存储器

虚拟存储器将主存或辅存的地址空间统一编址，形成一个庞大的存储空间。在这个大空间里，用户可以自由编程，完全不必考虑程序在主存是否装得下以及这些程序将来在主存中的实际存放位置。用户编程的地址称为虚地址或逻辑地址，实际的主存单元地址称为实地址或物理地址，虚地址空间要比实地址空间大得多。

虚拟存储器有页式、段式和段页式之分，其中最常用的是页式虚拟存储器。对于页式虚拟存储器，主存空间和虚存空间都划分成若干个大小相等的页，主存即实存的页称为实页，虚存的页称为虚页。虚地址到实地址之间的变换是由页表来实现的，页表是一张存放在主存中的虚页号和实页号的对照表，记录着程序的虚页调入主存时被安排在主存中的位置。使用页表进行地址转换的主要缺点是：每次访问存储器时都必须访问该页表。在带有单级页表的系统中，这样会使存储器的访问次数增加一倍，而在带有多级页表的系统中，该问题会变得更加严重，因为在遍历页表过程中需要进行多次存储器访问。

为了尽可能提高速度，可借鉴 Cache 的思路，将页表中最活跃的部分放在高速存储器中构成快表(TLB，又称为转换旁路缓冲器)，快表扮演的角色是作为页表的 Cache，对快表的查找和管理全用硬件来实现。快表一般很小，仅是主存中的页表(相对于快表称其为慢表)的一小部分。只有在快表中找不到(TLB 缺失)时，才去访问慢表。

5.3 典型例题详解

【例 5.1】 指令中地址码的位数与直接访问的存储器空间和最小寻址单位有什么关系？字编址计算机和字节编址计算机在地址码的安排上有何区别？PC 系列微机的指令系统可支持对字节、字、双字、四倍字的运算，试写出在对准边界时，字节地址、字地址、双字地址和四倍字地址的特点。

解： 主存容量越大，所需的地址码位数就越长；对于相同容量来说，最小寻址单位越小，地址码的位数就越长。

在一定容量的情况下，对于字编址的计算机，最小寻址单位是一个字，相邻的存储单元地址指向相邻的存储字，由于存储单元数目少，所以地址信息没有任何浪费。对于字节编址的计算机，最小寻址单位是一个字节，相邻的存储单元地址指向相邻的存储字节，由于存储单元数目多，所以地址信息存在着浪费。

PC 系列微机是一种字节编址的计算机，它支持字节(8 位)、字(16 位)、双字(32 位)和四倍字(64 位)的运算。不同宽度的数据存放在主存中，如果需要保证对准边界(即边界对齐)，则要求：字地址必须是 2 的整倍数，双字地址必须是 4 的整倍数，四倍字地址必须是 8 的整倍数。

【例 5.2】 设有一个 1MB 容量的存储器，字长为 32 位，问：

(1) 按字节编址，地址寄存器、数据寄存器各为几位？编址范围为多大？

(2) 按半字编址，地址寄存器、数据寄存器各为几位？编址范围为多大？

(3) 按字编址，地址寄存器、数据寄存器各为几位？编址范围为多大？

解： (1) 按字节编址，$1\text{MB}=2^{20}\times 8\text{b}$，地址寄存器为 20 位，数据寄存器为 8 位，编址范围为 00000H～FFFFFH。

(2) 按半字编址，$1\text{MB}=2^{20}\times 8\text{b}=2^{19}\times 16\text{b}$，地址寄存器为 19 位，数据寄存器为 16 位，编址范围为 00000H～7FFFFH。

(3) 按字编址，$1\text{MB}=2^{20}\times 8\text{b}=2^{18}\times 32\text{b}$，地址寄存器为 18 位，数据寄存器为 32 位，编址范围为 00000H～3FFFFH。

【例 5.3】 某机字长 32 位，主存储器按字节编址，现有 4 种不同长度的数据(字节、半字、单字、双字)，请采用一种既节省存储空间，又能保证任何长度的数据都在单个存取周期内完成读写的方法，将一批数据顺序地存入主存，画出主存中数据的存放示意图。

这批数据共有 10 个，依次为字节、半字、双字、单字、字节、单字、双字、半字、单字、字节。

解： 由于数据分成 4 种长度：字节(8 位)、半字(16 位)、单字(32 位)、双字(64 位)，任何长度的数据都能在单个存取周期内完成读写，故存储器的存储字长为 64 位。主存中的数据采用边界对齐的存放方法，10 个数据顺序存放的示意图如图 5-8 所示。

【例 5.4】 推算 16K×1 位双译码结构存储芯片的存储体阵列的行数和列数各是多少？若使用的存储芯片为动态 RAM，试求出该存储器的实际刷新时间(设刷新周期为 0.5μs)。

解： 16K×1 位存储芯片的存储阵列是行数和列数分别为 128 的方阵。

若使用的存储芯片为动态 RAM，则必须进行刷新，刷新是一行一行进行。所以该存储器的实际刷新时间为：

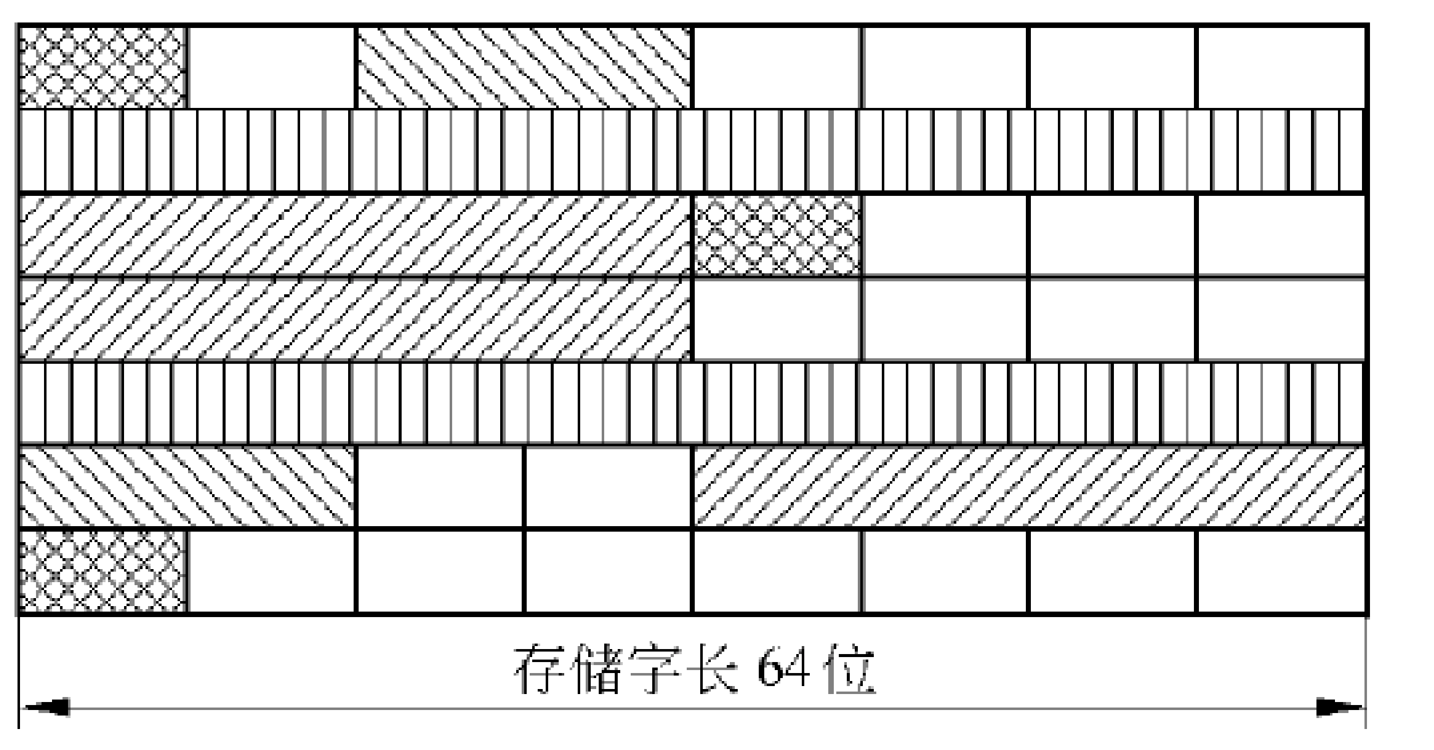


图 5-8 主存中数据的存放示意图

$$0.5 \times 128 = 64\mu s$$

【例 5.5】 图 5-9 是某 SRAM 的写入时序图，其中 $R/\overline{W}$ 是读写命令控制线，$R/\overline{W}$ 为低电平时，存储器按给定地址把数据线上的数据写入存储器。请指出图 5-9 中写入时序的错误，并画出正确的写入时序图。

解：在写入过程中，当 $R/\overline{W}$ 加负脉冲时，地址和数据线的电平必须是稳定的，否则将出现错误。当 $R/\overline{W}=0$ 时，如果数据线改变了数值，那么存储器将存储新的数据，(图 5-9 中的⑤)；如果地址线发生了变化，那么同样的数据将存储到前后两个地址中(图 5-9 的②和③)。正确的写入时序如图 5-10 所示。

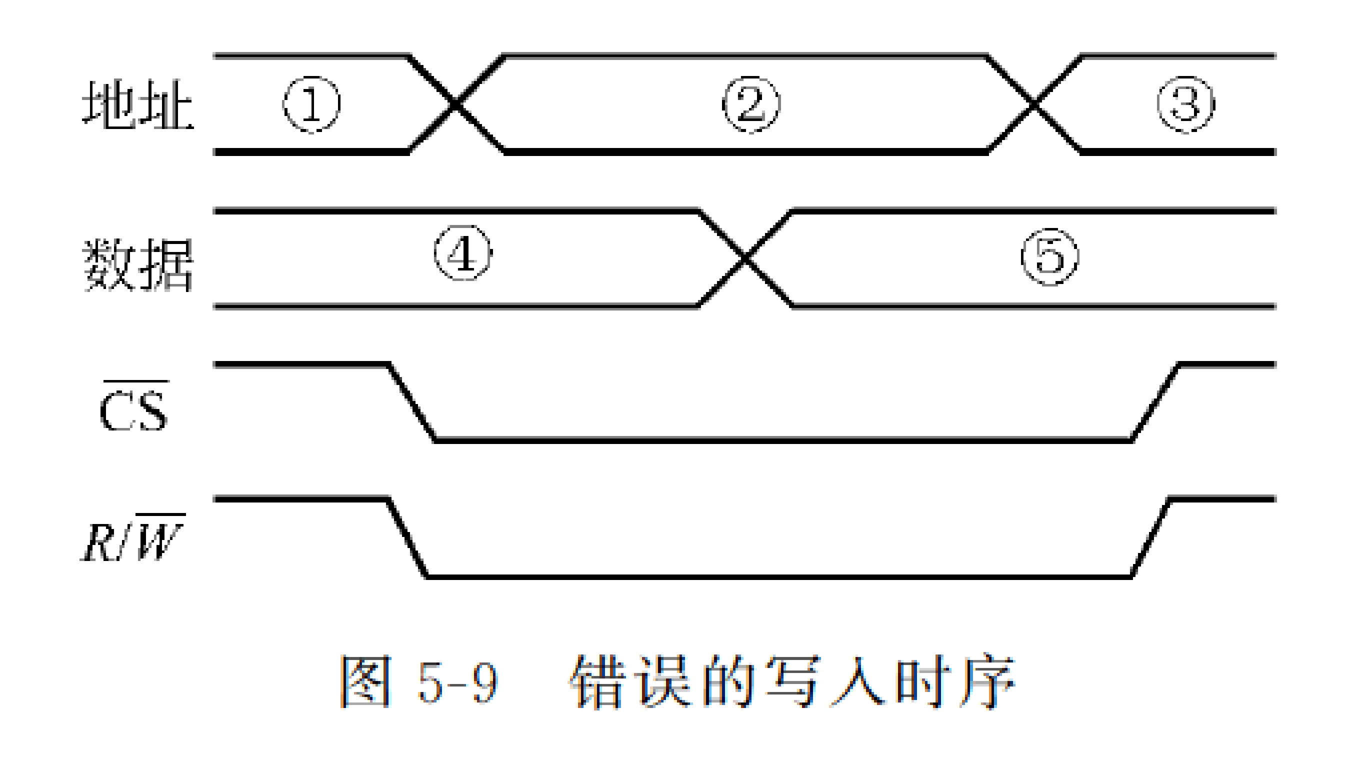


图 5-9 错误的写入时序

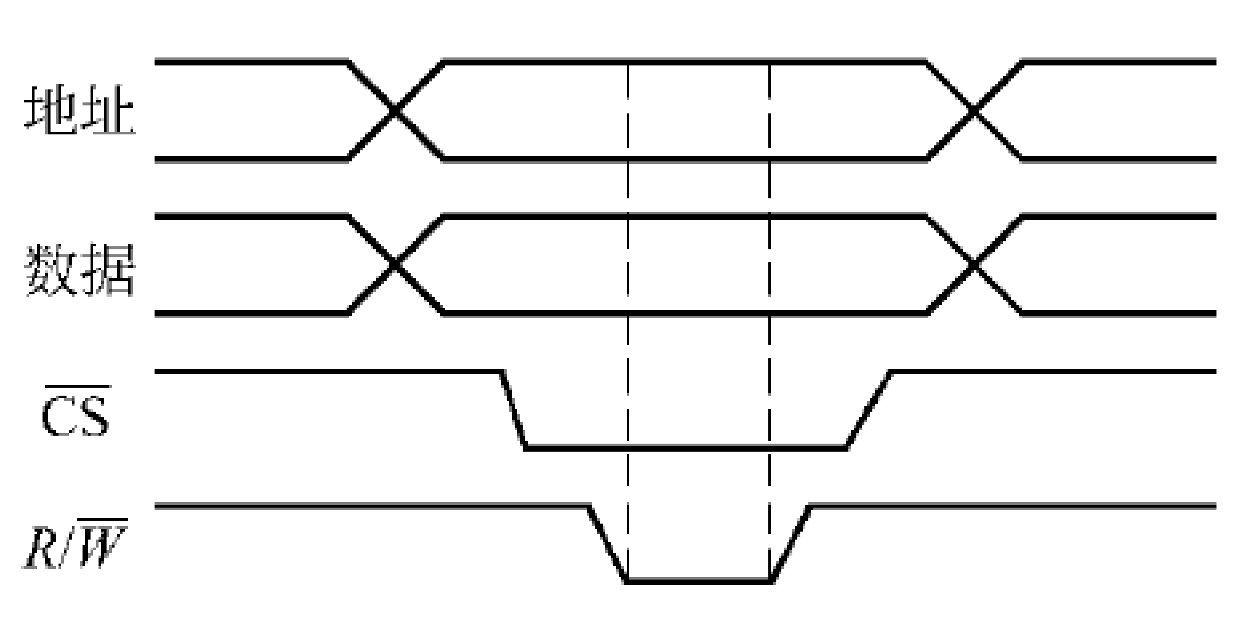


图 5-10 正确的写入时序

【例 5.6】 一台 8 位微机的地址总线为 16 条，其 RAM 存储器容量为 32KB，首地址为 4000H，且地址是连续的。问可用的最高地址是多少？

解：若 32KB 的存储地址起始单元为 0000H，则可知 32KB 存储空间共占用 15 条地址线，其范围应为 0000～7FFFH，但现在的首地址为 4000H，即首地址后移了，因此最高地址应为 4000H+7FFFH=BFFFH。

【例 5.7】 用容量为 $L \times K$ 的动态 RAM 芯片，构成容量为 $M \times N$ 的存储器。问：

(1) 需要多少块存储芯片？

(2) 存储器共有多少个片选信号，如何来实现？需要几位译码？

(3) 若采用自动刷新模式，刷新计数器的最大值是多少？

画出这个存储器的逻辑框图。

解：(1) 因为存储器的容量为 $M \times N$，存储芯片的容量为 $L \times K$，所以需要的存储芯片数为

$$\frac{M \times N}{L \times K}(\text{片})$$

(2) 这个存储器既使用了字扩展，又使用了位扩展。共有 $\frac{M}{L}$ 组存储芯片，需要 $\frac{M}{L}$ 个片

选信号。片选信号由译码器产生,需要 $\log_2\left\lceil\frac{M}{L}\right\rceil$ 位地址参与译码。

(3) 动态 RAM 需要刷新,刷新计数器的最大值是 $\sqrt{L\times K}$。这是因为在存储器中所有芯片同时被刷新,所以在考虑刷新问题时,应当从单个芯片的存储容量着手。在本题中动态 RAM 的内部结构应该是一个($\sqrt{L\times K}$)×($\sqrt{L\times K}$)的方阵,刷新通常是一行一行进行的,每一行中各记忆单元是同时被刷新的。这个存储器逻辑框图如图 5-11 所示。

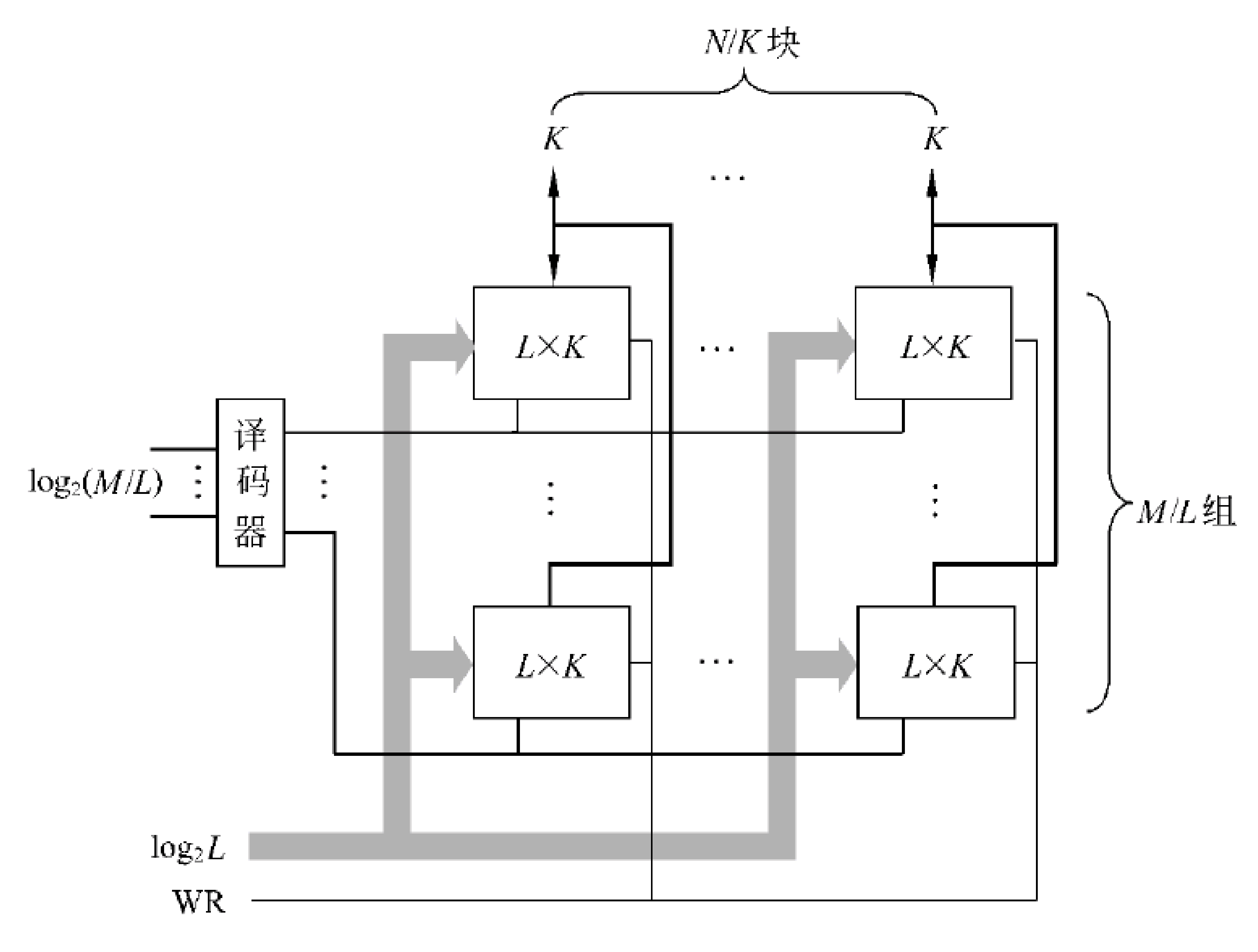


图 5-11 例 5.7 的存储器逻辑框图

【例 5.8】 通常存储芯片的容量是有限的,有时需要在字数和字长方面进行扩展。请用简单的例子说明,常用的 3 种扩展方法中,地址总线、数据总线、控制总线的连接规则以及所需的存储芯片数量。

解: 假定存储芯片容量为 $m\text{K}\times n$,容量扩展根据实际应用情况,可以有以下 3 种形式。

(1) 位扩展。例如要组成 $m\text{K}\times N$ 的存储器,需要 $\left\lceil\frac{N}{n}\right\rceil$ 个存储芯片。其连接结构中各芯片的地址、片选、写允许端都对应并接,数据输入、输出端则各自单独引出,即实现了位扩展。

(2) 字扩展。例如存储器的容量为 $M\text{K}\times n$,则需要 $\left\lceil\frac{M}{m}\right\rceil$ 个存储芯片。其连接结构中各芯片的地址、数据输入、数据输出、写允许端对应并接。片选信号单独引出,分别由存储器高 $\left\lceil\log_2\left\lceil\frac{M}{m}\right\rceil\right\rceil$ 位地址译码输出控制,在某一时刻只有一个片选信号有效。存储器的低 $\log_2 m\text{K}$ 位地址直接与芯片地址端连接。

(3) 字和位同时扩展。例如要组成 $M\text{K}\times N$ 的存储器,共需 $\left\lceil\frac{M}{m}\right\rceil\times\left\lceil\frac{N}{n}\right\rceil$ 个存储芯片。其连接结构中所有芯片写允许端并接,所有芯片地址端对应并接,直接连到存储器低 $\log_2 m\text{K}$ 位地址。同一行的片选端并接,行与行之间是独立的,分别由存储器高 $\left\lceil\log_2\left\lceil\frac{M}{m}\right\rceil\right\rceil$

位地址译码输出控制。输入、输出数据端同一列并接，列与列间是独立的。从纵向看，每列存储芯片给出不同存储单元的相同位；从横向看，每行存储芯片给出相同存储单元的不同位。

【例 5.9】 图 5-12 为用 8 片 2114 构成的 4K×8 的存储器，与 8 位的一个微处理器相连，2114 为 1024×4 位的静态 RAM 芯片。试求：

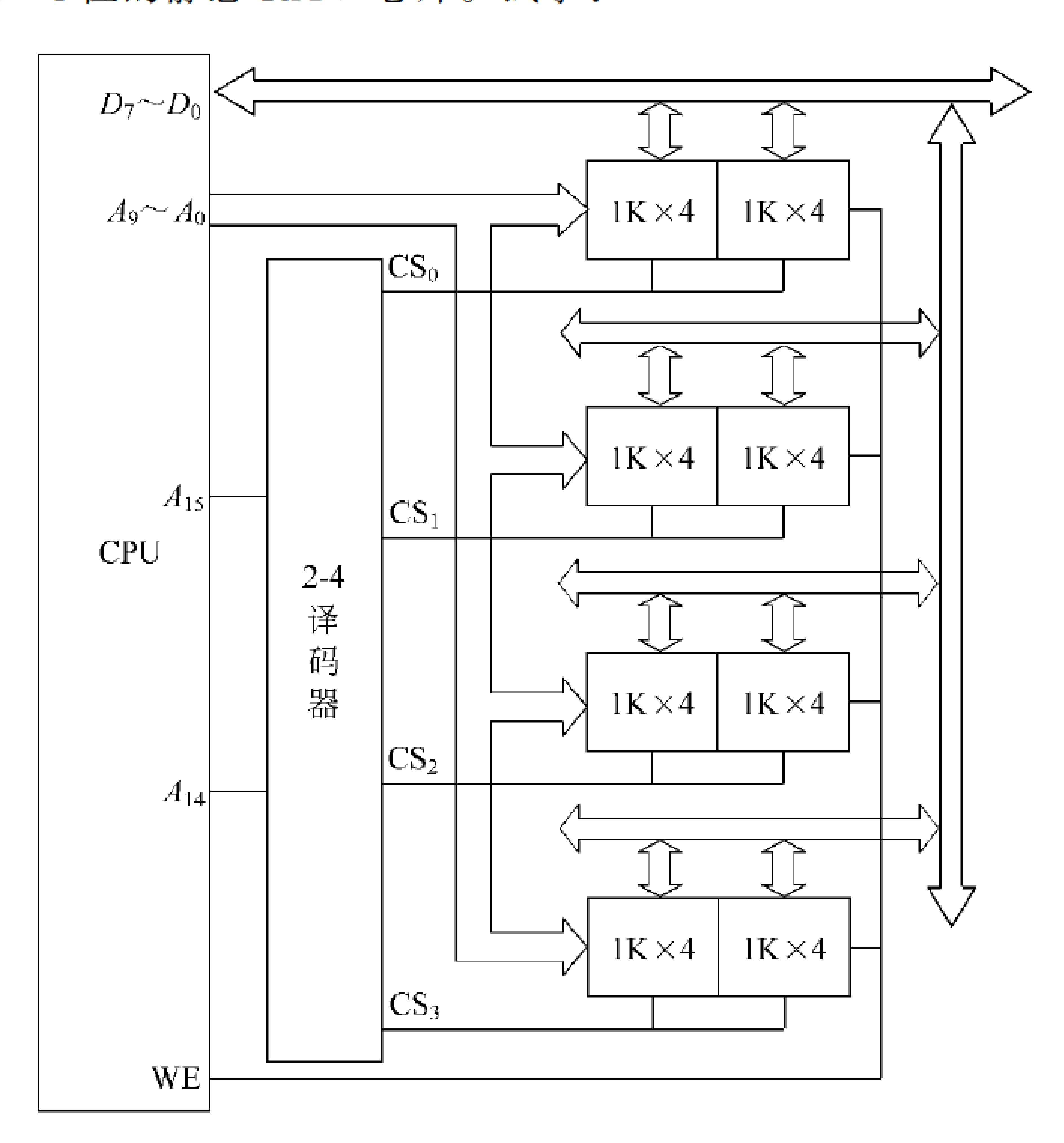


图 5-12　4K×8 的存储器与 CPU 的连接

(1) 每一组芯片组的地址范围、地址线数目。

(2) 4KB 的 RAM 寻址范围。

(3) 存储器有没有地址重叠？

解：(1) 芯片组的容量为 1024B，地址范围为 000H～3FFH，地址线数目 10 根(A_9～A_0)。

(2) 根据图 5-12 所示的连线，各芯片组的片选端由地址线 A_{15}、A_{14} 进行译码。芯片组内地址线为 A_9～A_0，A_{13}～A_{10} 空闲，即为任意态。假设 A_{13}～A_{10} 为全 0，4KB RAM 的寻址范围分别是：

第 0 组　0000H～03FFH

第 1 组　4000H～43FFH

第 2 组　8000H～83FFH

第 3 组　C000H～C3FFH

这 4KB 存储器的地址空间是不连续的。

(3) 由于 A_{13}～A_{10} 没有参与译码(部分译码)，所以存储器存在地址重叠现象。

【例 5.10】 图 5-13(a)所示为存储器的地址空间分布图，图 5-13(b)为存储器的地址译码电路，后者可在 A 组跨接端子和 B 组跨接端子之间进行连线，74LS139 是 2-4 译码器(A

为低端,B 为高端),使能端 $\overline{G}$ 接地表示译码器处于正常译码状态。

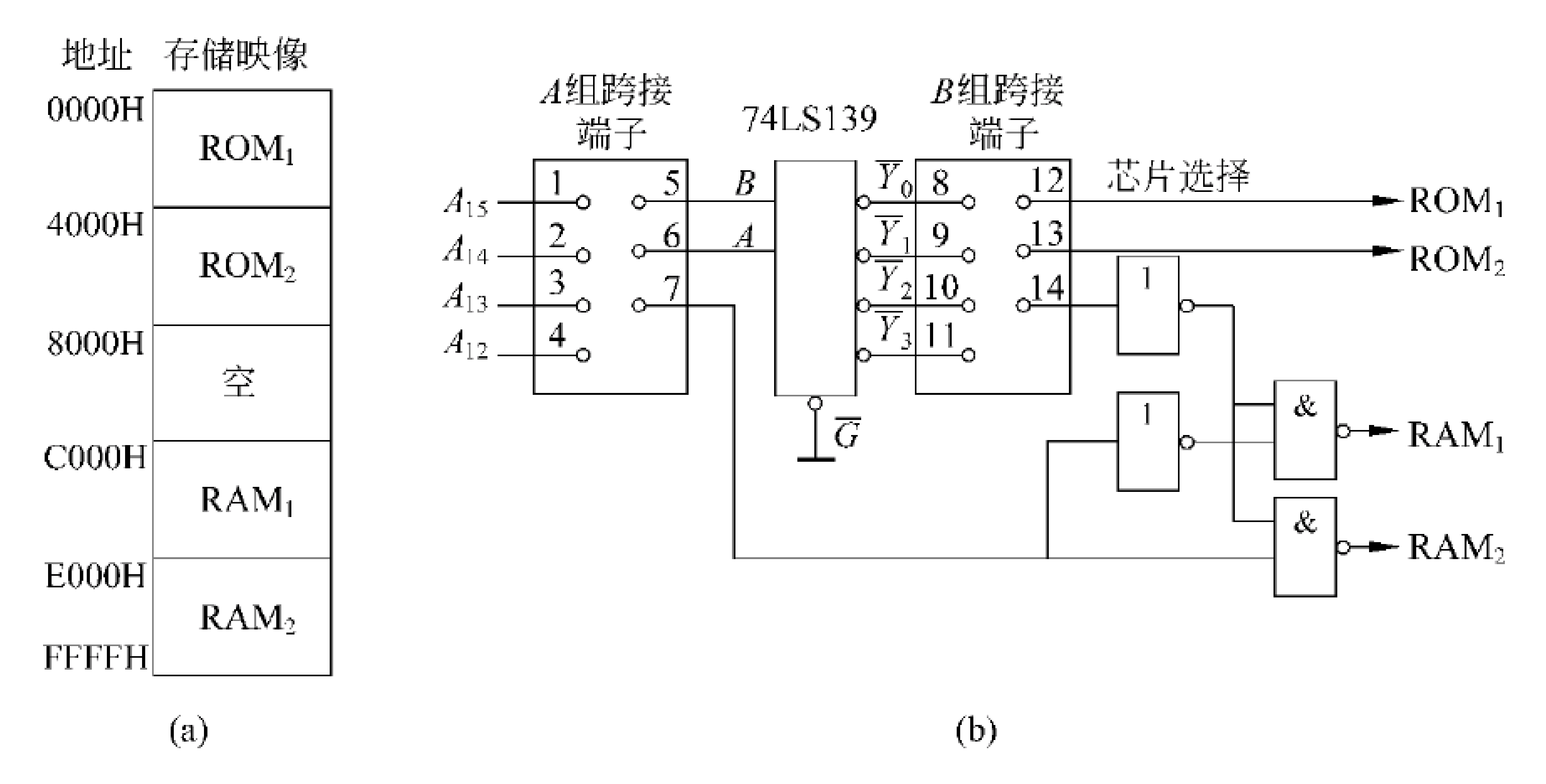


图 5-13　存储器的地址空间分布图和地址译码电路

要求:完成 A 组跨接端子和 B 组跨接端子内部的正确连接,以使地址译码器电路按图所示的要求进行正确寻址。

解:根据图 5-13(a)所示,可知各段占用的地址空间分别为:

ROM_1:0000H～3FFFH

ROM_2:4000H～7FFFH

RAM_1:C000H～DFFFH

RAM_2:E000H～FFFFH

对应上述地址空间,地址码最高 4 位 A_{15}～A_{12}的状态如下:

0000～0011　ROM_1

0100～0111　ROM_2

1100～1101　RAM_1

1110～1111　RAM_2

用 2-4 译码器 74LS139 对 $A_{15}A_{14}$ 两位进行译码,可产生 4 路输出。其中 $\overline{Y_0}$ 对应 ROM_1;$\overline{Y_1}$ 对应 ROM_2;$\overline{Y_2}$ 舍弃;$\overline{Y_3}$ 对应 RAM_1 和 RAM_2,然后将 A_{13} 分别取 0(对应 RAM_1)和 1(对应 RAM_2),再进行组合。由此两组端子的连接如下:

1—5,2—6,3—7,8—12,9—13,11—14。

【例 5.11】 某计算机系统中 CPU 可输出 20 条地址线(A_{19}～A_0),8 条数据线(D_7～D_0)和一条控制线($\overline{WE}$),主存按字节编址,由 16KB 的 ROM 和 64KB 的 RAM 组成。拟采用 8K×4 的 ROM 芯片 2 片,8K×8 的 ROM 芯片 1 片,32K×2 的 RAM 芯片 4 片,32K×8 的 RAM 芯片 1 片。

解:该系统中由于对主存地址范围无特殊要求,而且实际使用的主存空间比 CPU 可访问的最大存储空间小得多,因此主存可采用部分译码方式,其连接图如图 5-14 所示。

从图 5-14 中可以看出,地址码 A_{19}、A_{18}、A_{17} 没有参加译码,对于芯片①②③来说,还有 A_{14}、A_{13} 也没有参加译码,A_{16}、A_{15} 经 2-4 译码器译码后产生 4 个片选信号,分别选择 4 组芯片,于是各组芯片的地址分配如表 5-2 所示。

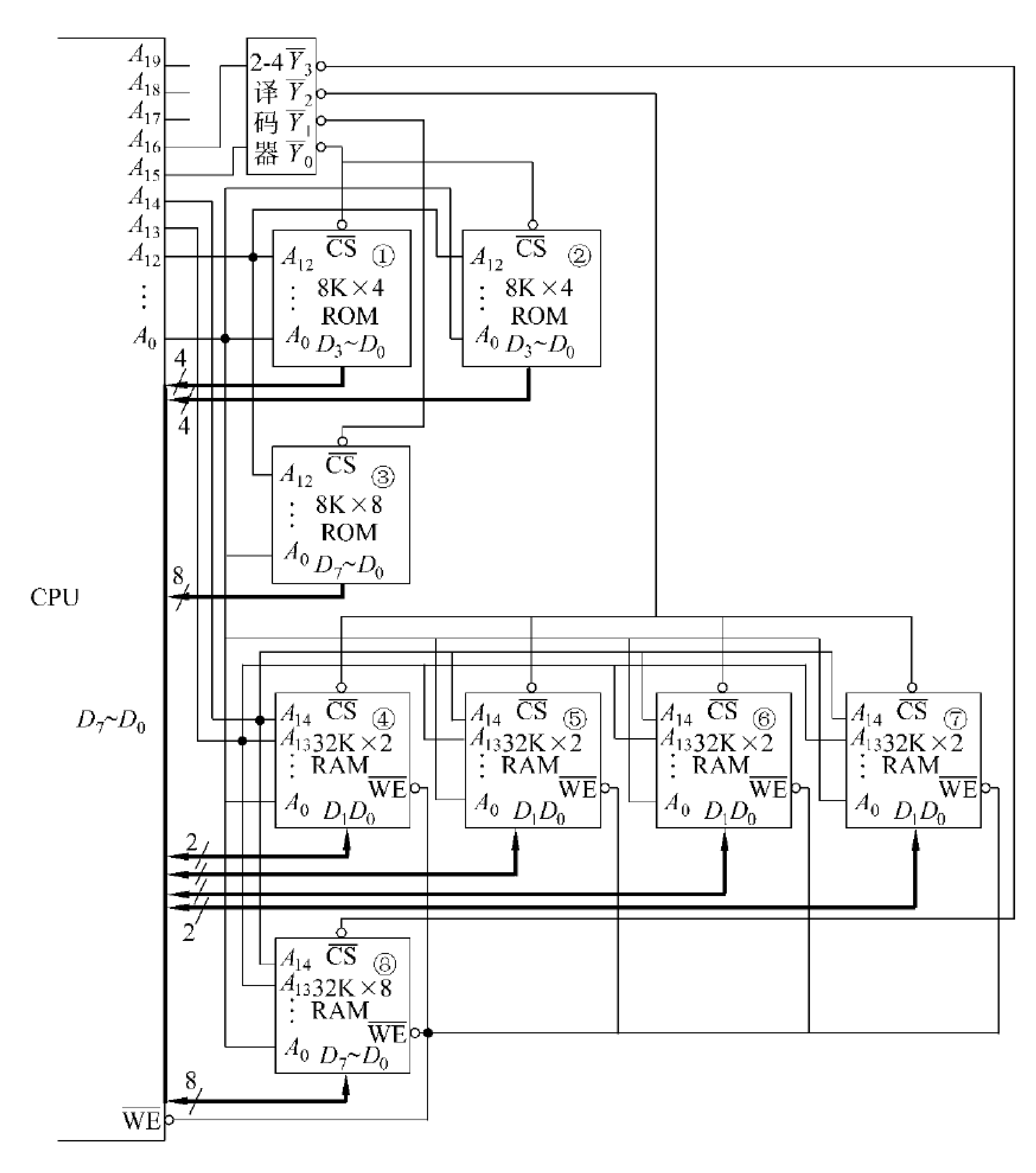


图 5-14　主存与 CPU 的连接方式之一

表 5-2　主存储器地址分配

所选芯片	选片地址							片内地址					译码输出	地址分配	重叠区个数
	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	…	A_0			
①②	×	×	×	0	0	×	×	0	0	0	…	0	$\overline{Y}_0=0$	00000H ~01FFFH	32 个
	×	×	×	0	0	×	×	1	1	1	…	1			
③	×	×	×	0	1	×	×	0	0	0	…	0	$\overline{Y}_1=0$	08000H ~09FFFH	32 个
	×	×	×	0	1	×	×	1	1	1	…	1			
④⑤⑥⑦	×	×	×	1	0	0	0	0	0	0	…	0	$\overline{Y}_2=0$	10000H ~17FFFH	8 个
	×	×	×	1	0	1	1	1	1	1	…	1			
⑧	×	×	×	1	1	0	0	0	0	0	…	0	$\overline{Y}_3=0$	18000H ~1FFFFH	8 个
	×	×	×	1	1	1	1	1	1	1	…	1			

表 5-2 中各个地址范围都不是唯一的，例如芯片①②在表中所示的地址范围为 00000H～01FFFH，这仅仅是将未参加译码的地址 A_{19}、A_{18}、A_{17}、A_{14}、A_{13}全部取 0 所得到的，实际上这 5 位地址可构成 32 种组合，都会选上这 2 个芯片，因此，对于芯片①②，可构成 32 个地址重

叠区。

其他组芯片的情况类似,总之,出现多个地址重叠区的原因是由于有一部分地址码没有参加译码。

【例 5.12】 上例中给定的条件不变,只是要求每组芯片有唯一的地址分配(即无地址重叠区),其具体要求如下:

①②号芯片的地址范围为 18000H~19FFFH。

③号芯片的地址范围为 F8000H~F9FFFH。

④~⑦号芯片的地址范围为 98000H~9FFFFH。

⑧号芯片的地址范围为 B8000H~BFFFFH。

解: 由于每组芯片都有唯一的地址分配,必须采用全译码方式,其连接图如图 5-15 所示。

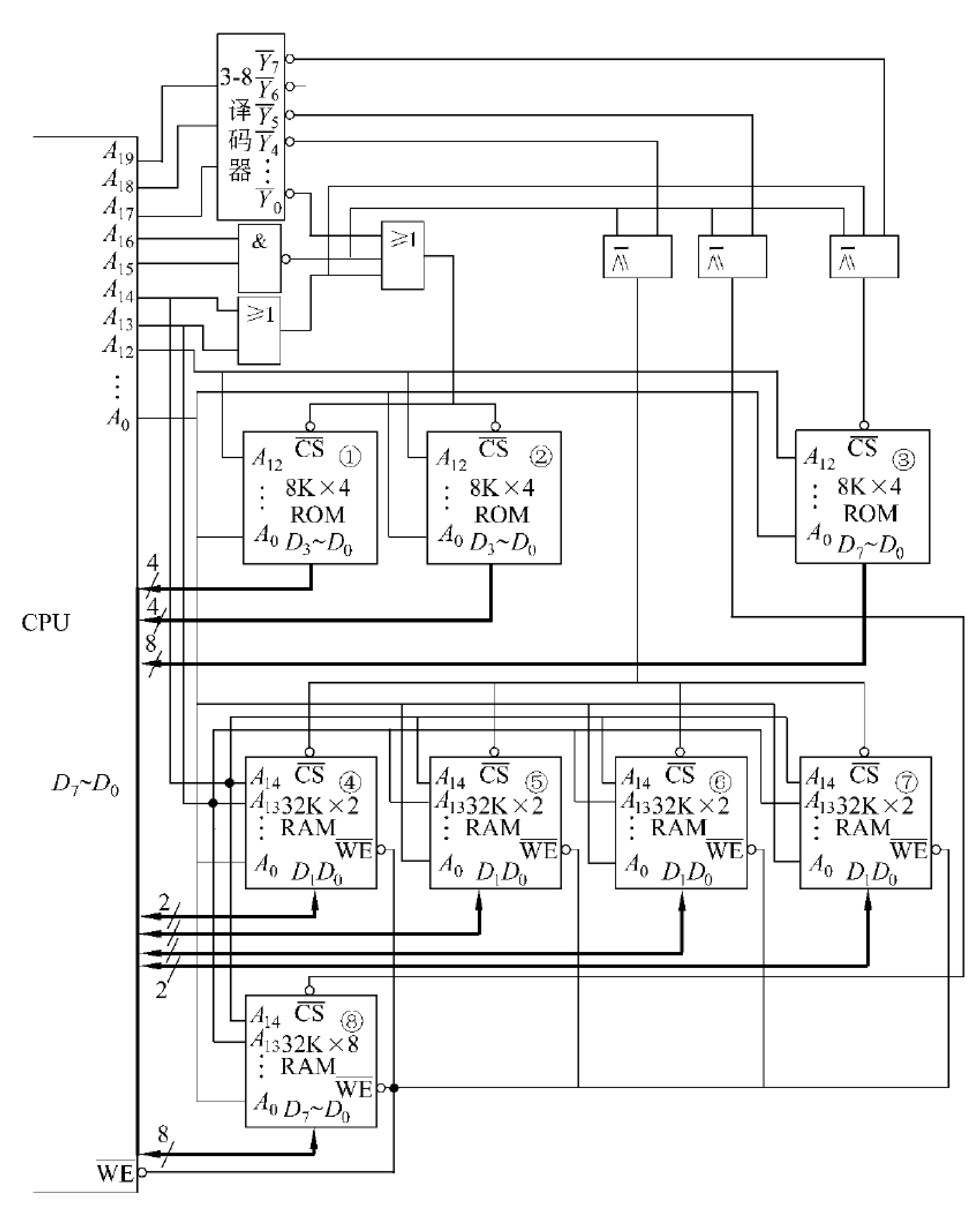


图 5-15 主存与 CPU 的连接方式之二

从图 5-15 中可以看出,采用全译码方式时,由于对某些位地址码有固定置“1”和置“0”的要求,因此除了直接参加译码的 3 位地址码(A_{19}、A_{18}、A_{17})之外,还需要增设一些逻辑门

以满足对地址码 A_{16}、A_{15} 固定置“1”的要求和对 A_{14}、A_{13} 固定置“0”的要求。于是各组芯片的地址分配如表 5-3 所示。

表 5-3　主存储器地址分配

所选芯片	选片地址							片内地址					译码输出	地址分配
	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	…	A_0		
①②	0	0	0	1	1	0	0	0	0	0	…	0	$\overline{Y}_0=0$	18000H～19FFFH
	0	0	0	1	1	0	0	1	1	1	…	1		
④⑤⑥⑦	1	0	0	1	1	0	0	0	0	0	…	0	$\overline{Y}_4=0$	98000H～9FFFFH
	1	0	0	1	1	1	1	1	1	1	…	1		
⑧	1	0	1	1	1	0	0	0	0	0	…	0	$\overline{Y}_5=0$	B8000H～BFFFFH
	1	0	1	1	1	1	1	1	1	1	…	1		
③	1	1	1	1	1	0	0	0	0	0	…	0	$\overline{Y}_7=0$	F8000H～F9FFFH
	1	1	1	1	1	0	0	1	1	1	…	1		

【例 5.13】 用 2K×8 的芯片设计一个 8K×16 的存储器：当 $B=0$ 时，访问 16 位数；当 $B=1$ 时，访问 8 位数。

解：由于要求存储器能按字节访问，即 $8\text{K}\times16=16\text{K}\times8=2^{14}\times8$，所以地址线需14 根，数据线需 16 根。

先设计一个模块将 2K×8 扩展成 2K×16，内部地址为 A_{11}～A_1。设计方案如图 5-16 和表 5-4 所示。

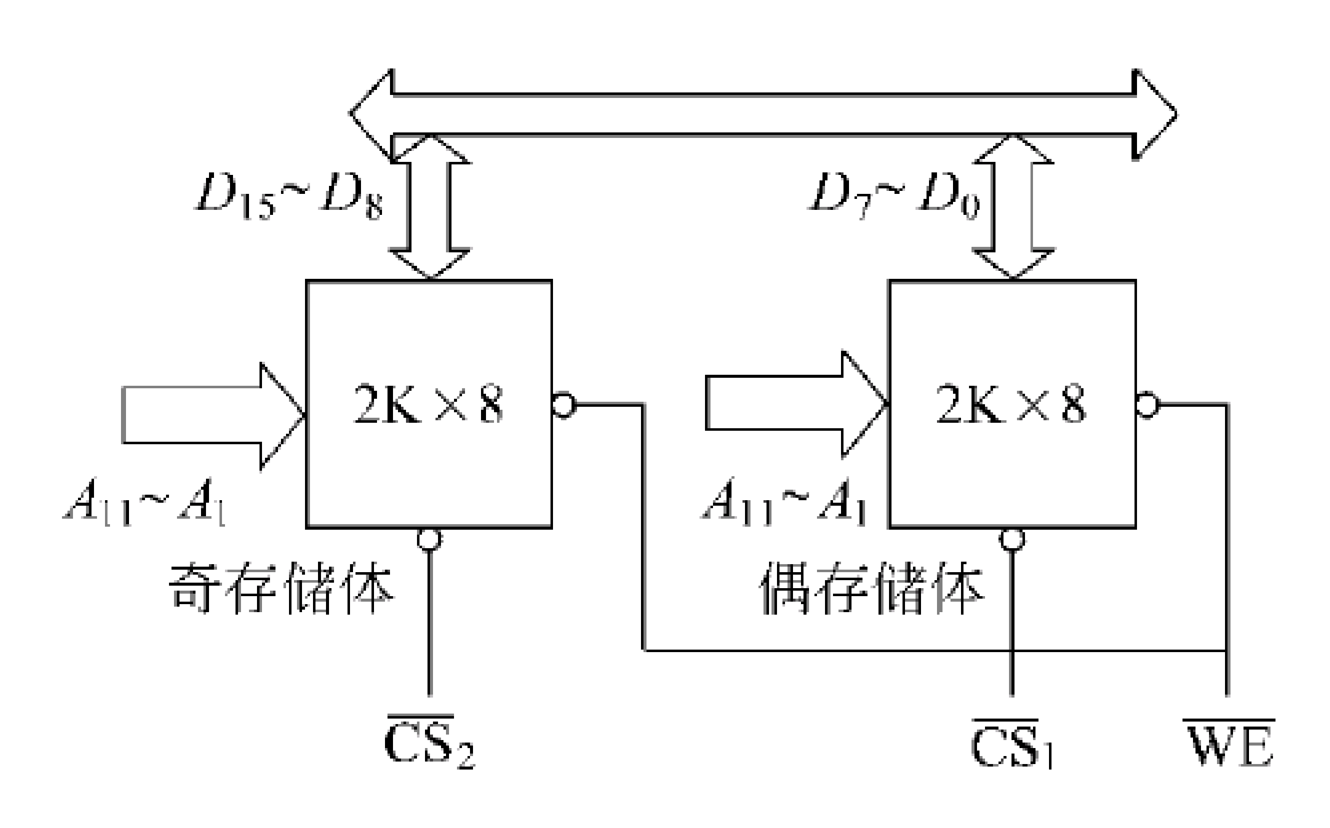


图 5-16　2K×8 扩展成 2K×16 的模块

表 5-4　访存控制信号安排

B	A_0	C	D	说　明
0	0	1	1	访问 16 位数
0	1	0	0	不访问
1	0	1	0	访问偶存储体
1	1	0	1	访问奇存储体

8K×16 的存储器需要 4 个模块，因此需要用 2-4 译码器，设译码器的输出分别为 Y_0、Y_1、Y_2、Y_3，则 $\overline{CS_1}$、$\overline{CS_2}$、$\overline{CS_3}$、$\overline{CS_4}$、$\overline{CS_5}$、$\overline{CS_6}$、$\overline{CS_7}$、$\overline{CS_8}$ 的表达式分别为：

$$\overline{CS_1}=\overline{C\cdot Y_0}\quad \overline{CS_2}=\overline{D\cdot Y_0}$$
$$\overline{CS_3}=\overline{C\cdot Y_1}\quad \overline{CS_4}=\overline{D\cdot Y_1}$$
$$\overline{CS_5}=\overline{C\cdot Y_2}\quad \overline{CS_6}=\overline{D\cdot Y_2}$$
$$\overline{CS_7}=\overline{C\cdot Y_3}\quad \overline{CS_8}=\overline{D\cdot Y_3}$$

存储器结构图及与 CPU 连接的示意图如图 5-17 所示。

【例 5.14】 有 4 片 Intel 2114 芯片，按图 5-18 所示连接。试问：

(1) 图 5-18 所示的连接组成几部分存储区域？共有多大的存储器容量？字长是多少？

(2) 写出每部分存储区域的地址范围。



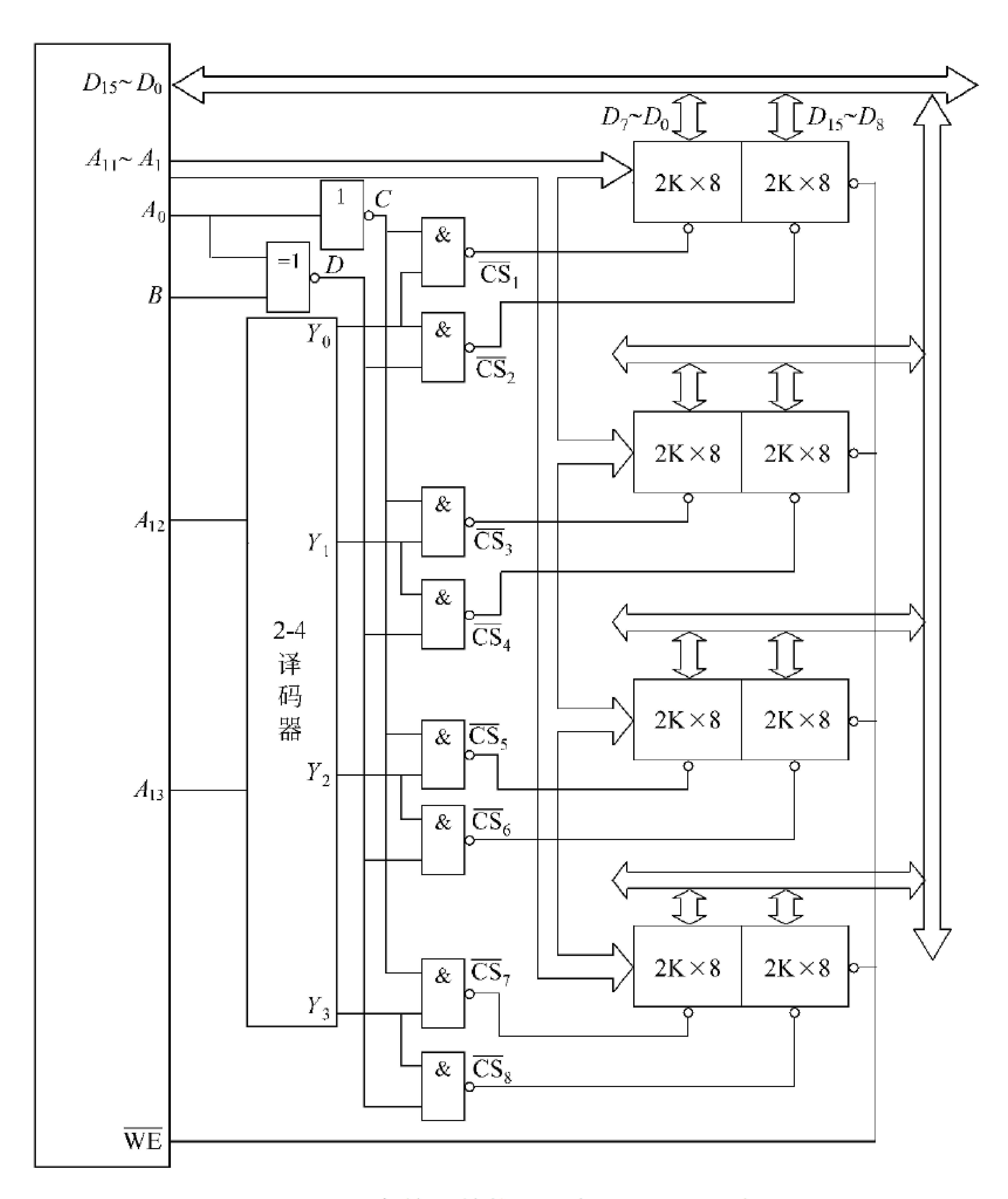


图 5-17　存储器结构图及与 CPU 的连接

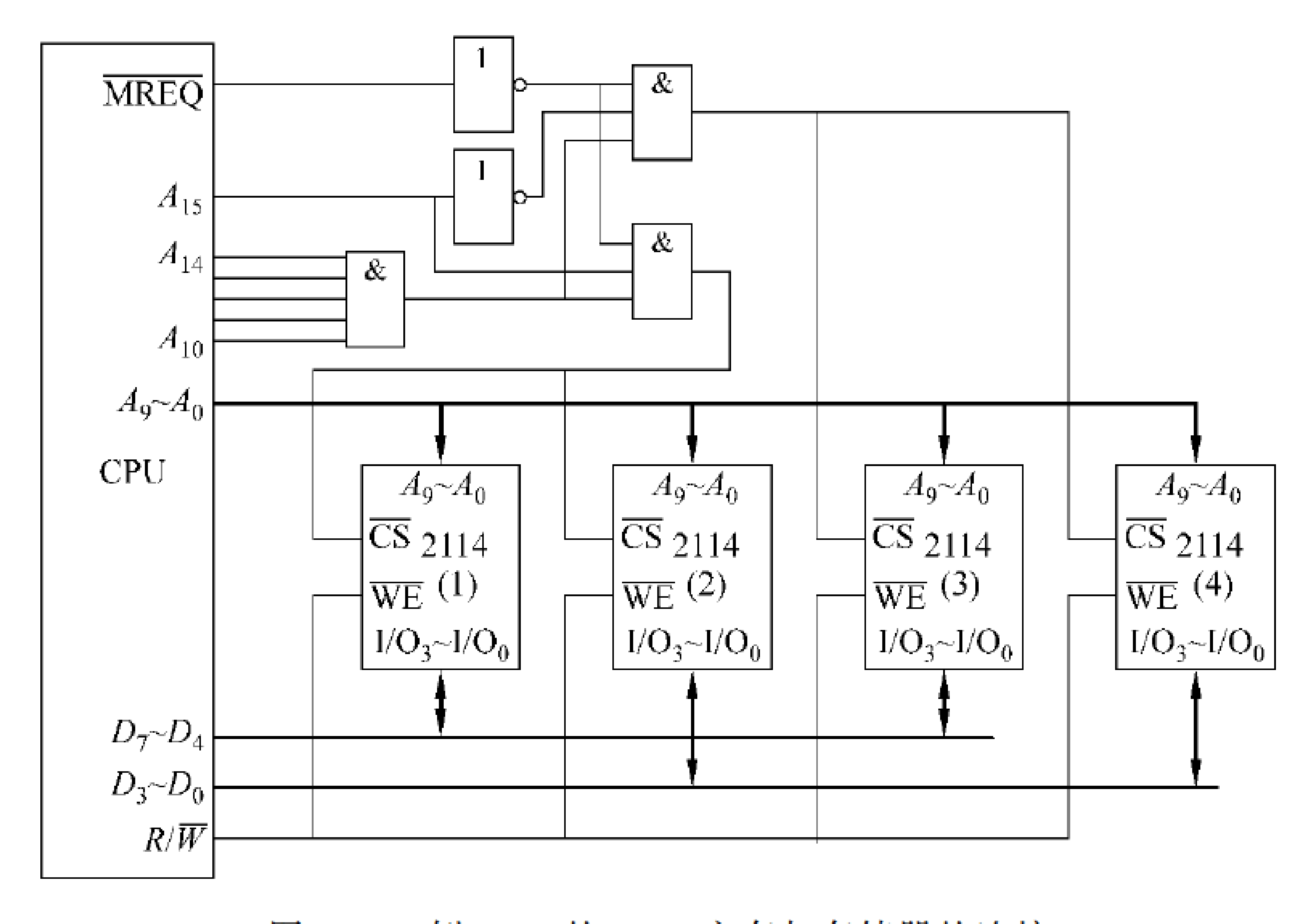


图 5-18　例 5.14 的 CPU 主存与存储器的连接

(3) 说明图中存储器的地址是否连续。若不连续,应该怎样修改才能使存储器的地址是连续的?

解:(1) 2114 芯片的容量为 1K×4,4 片 2114 分为两组,组内位扩展,组间字扩展,组成 2K×8 的存储容量,字长 8 位。

(2) 根据图 5-18 的片选逻辑电路,可以看出第一组的地址范围是 FC00H~FFFFH,第二组的地址范围 7C00H~7FFFH,见表 5-5。

表 5-5 存储器的地址范围

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	$A_9 \sim A_0$	地址范围
第一组 (1)(2)	1	1	1	1	1	1	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	FC00H FFFFH
第二组 (3)(4)	0	1	1	1	1	1	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	7C00H 7FFFH

(3) 从表 5-5 中可以看出存储器的地址是不连续的,若修改片选电路,将 A_{15} 与 A_{10} 对换,可以使得存储器的地址连续,见表 5-6。

表 5-6 变换后存储器的地址范围

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	$A_9 \sim A_0$	地址范围
第一组 (1)(2)	1	1	1	1	1	1	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	FC00H FFFFH
第二组 (3)(4)	1	1	1	1	1	0	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	F800H FBFFH

【例 5.15】 已知地址总线为 $A_{15} \sim A_0$(A_0 为最低位),数据总线为 $D_7 \sim D_0$。用 1 片 16K×8 的 RAM 芯片(地址从 0000H 开始)、2 片 8K×8 的 RAM 芯片(地址从 4000H 开始)、4 片 2K×4 的芯片(地址从 8000H 开始),将上述芯片组成一个存储器,片选信号均为低电平有效,该存储器按字节编址,假设读写信号是 $R/\overline{W}$,不考虑存储器刷新。

(1) 为各芯片分配地址空间;

(2) 说明各芯片需要多少条地址线;

(3) 写出各芯片的片选信号逻辑表达式;

(4) 画出存储器的逻辑电路图。

解:(1) 各芯片的地址范围如下:

① 16K×8 RAM　　0000H~3FFFH

② 8K×8 RAM　　4000H~5FFFH

③ 8K×8 RAM　　6000H~7FFFH

④⑤ 2K×4 RAM　　8000H~87FFH

⑥⑦ 2K×4 RAM　　8800H~8FFFH

(2) 16K×8 的 RAM 芯片需要 14 条地址线,8K×8 的 RAM 芯片需要 13 条地址线,2K×4 的 RAM 芯片需要 11 条地址线。

(3) 5个片选信号的逻辑表达式如下:

$$CS_0 = \overline{A_{15}}\ \overline{A_{14}}$$

$$CS_1 = \overline{A_{15}}A_{14}\ \overline{A_{13}}$$

$$CS_2 = \overline{A_{15}}A_{14}A_{13}$$

$$CS_3 = A_{15}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}\ \overline{A_{11}}$$

$$CS_4 = A_{15}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}A_{11}$$

(4) 存储器的逻辑电路图如图5-19所示。

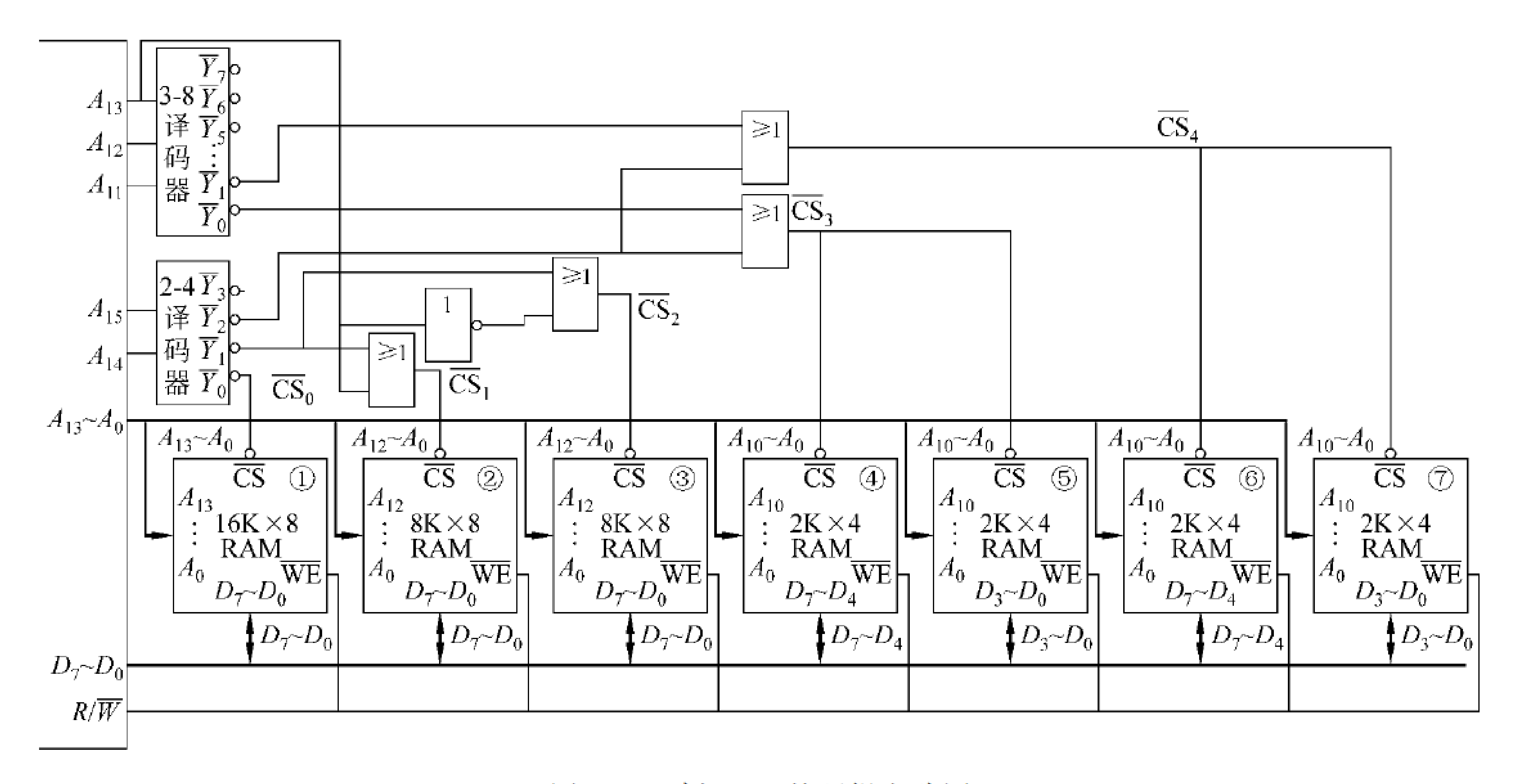


图5-19 例5.15的逻辑电路图

【例5.16】 有一个16K×6的存储器,由1K×4的DRAM芯片(内部结构为64×64)构成,问:

(1) 采用异步刷新方式,如最大刷新间隔为2ms,则相邻两行之间的刷新间隔是多少?

(2) 如采用集中刷新方式,存储器刷新一遍最少用多少个刷新周期?设存储器的读写周期为0.5μs,死区占多少时间?死时间率为多少?

解:(1) 采用异步刷新方式,在2ms时间内把芯片的64行刷新一遍,相邻两行之间的刷新间隔=2ms÷64=31.25μs,可取的刷新间隔为31μs。

(2) 如采用集中刷新方式,存储器刷新一遍最少用64个刷新周期,因为存储器的读写周期为0.5μs,刷新周期也为0.5μs,死区=0.5μs×64=32μs,死时间率=32μs÷2000μs×100%=1.6%。

【例5.17】 设有4个模块组成的4体存储器结构,每个存储体的存储字长为32位,存储周期为200ns。假设数据总线宽度为32位,总线传输周期为50ns,试求顺序存储(高位交叉)和交叉存储(低位交叉)的存储器带宽。

解:顺序存储存储器连续读出4个字的时间是200ns×4=800ns=8×10^{-7}s。

交叉存储存储器连续读出4个字的时间是200ns+50ns×(4−1)=350ns=3.5×10^{-7}s。

顺序存储存储器的带宽是 $128 \div (8\times10^{-7}) = 16\times10^{7}$ b/s。

交叉存储存储器的带宽是 $128 \div (3.5\times10^{-7}) = 37\times10^{7}$ b/s。

【例 5.18】 Cache 存取周期为 45ns，主存存取周期为 200ns。已知在一段给定的时间内，CPU 共访存 4500 次，而 Cache 的未命中率为 10%，问：

(1) CPU 访问 Cache 和主存各多少次？

(2) CPU 访存的平均访问时间是多少？

(3) Cache-主存系统的效率是多少？

解：(1) CPU 共访存 4500 次，Cache 未命中 10%，需要访问主存，访问主存次数 = 4500×10% = 450 次，则访问 Cache 次数 = 4500 − 450 = 4050 次。

(2) $T_A = H\times T_{A_1} + (1-H)\times T_{A_2} = 0.9\times45\text{ns} + 0.1\times200\text{ns} = 60.5\text{ns}$。

(3) $e = \dfrac{T_{A_1}}{T_A} = \dfrac{45}{60.5} \approx 74.4\%$。

【例 5.19】 设计算机的存储器为 64K×16 位，直接地址映像的 Cache 容量为 1KW (KW，千字)，每块 4 字。

(1) Cache 地址的标志字段、块号和块内地址分别有多少位？

(2) Cache 中可装入多少块数据？

解：(1) 主存的容量为 64KW，每字 16 位。

$64\text{K} = 2^{16}$，主存字地址有 16 位；$1\text{K} = 2^{10}$，Cache 字地址有 10 位。

$4 = 2^2$，块内地址 2 位，块号有(10−2)位 = 8 位。

标志字段(16−10)位 = 6 位。

(2) $2^8 = 256$，Cache 中可装入 256 块数据。

【例 5.20】 假设主存容量为 512KB，Cache 容量为 4KB，每个字块为 16 个字，每个字 32 位，按字节编址。

(1) Cache 地址有多少位？可容纳多少块？

(2) 主存地址多少位？可容纳多少块？

(3) 在直接映射方式下，主存的第几块映射到 Cache 的第 5 块(设起始字块为第 1 块)？

(4) 画出直接映射方式下主存地址字段中各段的位数。

解：(1) 根据 Cache 容量 4KB，Cache 地址为 12 位。由于每字 32 位，按字节编址，每个字为 4 个字节，故 Cache 共有 1KW(4KB÷4B)。又因为每个字块 16 个字(W)，故 Cache 中有 64 个字块(1K÷16)。

(2) 根据主存容量 512KB，主存地址 19 位。由于每字 32 位，故主存共有 128K 字(512KB÷4B)。又因为每个字块 16 个字，故主存中共有 8 192 个字块(128K÷16)。

(3) 直接映射将主存中的每一个块放置到 Cache 中唯一的一个指定位置上，主存的 5，64+5，2×64+5，…，$2^{13}\times64+5$ 能映射到 Cache 的第 5 块中。

(4) 字块内地址 6 位，Cache 字块地址 6 位，主存字块标记 7 位(主存地址长度与 Cache 地址长度之差)。主存地址字段的各段位数如图 5-20 所示。

主存标记	Cache块号	块内地址
7位	6位	6位

图 5-20　例 5.20 的主存地址格式

【例 5.21】 设某机主存容量为 16MB，Cache 的容量为 8KB。每字块 8 个字，每字

32位。设计一个四路组相联映射的Cache组织。

(1) 画出主存地址字段中各段的位数。

(2) 设Cache初态为空,CPU依次从主存0,1,2,…,99号单元中读出100个字(主存一次读出一个字),并重复此次序10次,问命中率是多少?

(3) 若Cache速度是主存速度的5倍,试问有Cache和无Cache相比,速度提高多少倍?

(4) 系统的效率是多少?

解:(1) 主存地址字段如图5-21所示。根据每个字32位(4B),每个字块8个字,得出块内地址5位。根据Cache容量8KB,字块大小32B,得出Cache共256块。根据4路组相联,共分64组(256÷4)。

主存标记	组号	块内地址
13位	6位	5位

图5-21　例5.21的主存地址格式

(2) 由于Cache初态为空,CPU读0号单元时不命中,必须访存,同时将该字所在的主存块调入Cache,接着CPU读1~7号单元均命中。同理,CPU读8,16,…,96号单元均不命中。可见CPU在连续读100个字中共有13次未命中,而后9次循环读100个字全部命中,命中率为

$$\frac{100\times 10-13}{100\times 10}\times 100\% = 98.7\%$$

(3) 设主存存取周期为$5t$,Cache存取周期为t,没有Cache的访问时间是$5t\times 1000$,Cache的访问时间是$t\times(1000-13)+5t\times 13$,则有Cache和无Cache相比,速度提高的倍数为

$$\frac{5t\times 1000}{t\times(1000-13)+5t\times 13}-1\approx 3.75$$

(4) 系统的效率为

$$\frac{t}{0.987\times t+(1-0.987)\times 5t}\times 100\% = 95\%$$

【例5.22】 某计算机的主存地址位数为32位,按字节编址。假定数据Cache中最多存放128个主存块,采用4路组相联方式,块大小为64B,每块设置了1位有效位。采用一次性写回策略,为此每块设置了1位“脏”位。要求:

(1) 分别指出主存地址中标记(Tag)、组号(Index)和块内地址(Offset)3部分的位置和位数。

(2) 计算该数据Cache的总位数。

解:(1) 因为块大小为64B,所以块内地址字段为6位,位于主存地址后部;因为Cache中有128个主存块,采用4路组相联,Cache分为32组(128÷4=32),所以组号字段为5位,位于主存地址中部;标记字段为剩余位,32-5-6=21位,位于主存地址前部。

(2) 数据Cache的总位数应该包括标记项的总位数和数据块的位数。每个Cache块对应一个标记项,标记项中应包括标记字段、有效位和脏位(仅适用于写回法)。故标记项的总位数=128×(21+1+1)=128×23=2944位。数据块位数=128×64×8=65 536位,所以数据Cache的总位数=2944+65 536=68 480位。

【例5.23】 某个系统拥有48位的虚拟地址和36位的物理地址,并且主存储器的容量为128MB。如果系统中使用的页的大小为4096B,问:该地址空间能够支持的虚页数和实

页数分别是多少？主存储器中共有多少个页框？

解：$4096=2^{12}$，所以虚拟地址和物理地址中的低12位被用作页内地址字段，虚页号的长度为 48－12＝36，所以虚拟地址空间能支持 2^{36} 个虚页；而实页号的长度为 36－12＝24，所以物理地址空间能支持 2^{24} 个实页。主存储器的页框数即主存中可同时包含的页数，即 128MB÷4KB＝32 768。

*【例 5.24】 某计算机的 Cache 共有 16 块，采用 2 路组相联映射方式（即每组 2 块）。每个主存块大小为 32B，按字节编址。主存 129 号单元所在主存块应装入到的 Cache 组号是________。

A. 0　　B. 2　　C. 4　　D. 6

分析：由于每个主存块大小为 32B，按字节编址。根据计算主存块号的公式，主存块号$=\lfloor$主存地址/块大小$\rfloor=\left\lfloor\frac{129}{32}\right\rfloor=4$，所以主存 129 号单元所在的主存块应为第 4 块。若 Cache 共有 16 块，采用 2 路组相联映射方式，可分为 8 组。根据组相联映像的映射关系，主存第 4 块转入 Cache 第 4 组。

目前对于组相联具体映射方法，在不同的书上有不同的说法，详见本章重点难点梳理 17，所以这道题的选项 B 也可以认为是正确的。其主要区别在于主存地址字段上，若主存地址被分为标记、组号、块内地址 3 字段结构，正确答案是 C；若主存地址被分为区号、组号、组内块号、块内地址 4 字段结构，正确答案是 B。

*【例 5.25】 某计算机主存容量为 64KB，其中 ROM 区为 4KB，其余为 RAM 区，按字节编址。现要用 2K×8 的 ROM 芯片和 4K×4 的 RAM 芯片来设计该存储器，则需要上述规格的 ROM 芯片数和 RAM 芯片数分别是________。

A. 1、15　　B. 2、15　　C. 1、30　　D. 2、30

解：D。

分析：ROM 区为 4KB，选用 2K×8 的 ROM 芯片，需要 2 片，采用字扩展方式；60KB 的 RAM 区，选用 4K×4 的 RAM 芯片，需要 30 片，采用字和位同时扩展方式。

*【例 5.26】 假设某计算机的存储系统由 Cache 和主存组成。某程序执行过程中访存 1000 次，其中访问 Cache 缺失（未命中）50 次，则 Cache 的命中率是________。

A. 5%　　B. 9.5%　　C. 50%　　D. 95%

解：D。

分析：Cache 的命中率 $H=\frac{N_1}{N_1+N_2}$，程序访存次数（包括访问 Cache 的次数和访存主存的次数）为＝1000 次，其中访问 Cache 的次数 N_1 为访存次数减去失效次数（1000－50＝950）。所以 $H=\frac{1000-50}{1000}=0.95=95\%$。

*【例 5.27】 假定用若干个 2K×4 的芯片组成一个 8K×8 的存储器，则地址 0B1FH 所在芯片的最小地址是

A. 0000H　　B. 0600H　　C. 0700H　　D. 0800H

解：D。

分析：由若干芯片构成存储器，采用字和位同时扩展方法。8 片 2K×4 的芯片分成

4组,每组2个芯片,各组芯片的地址分配分别为:第1组,0000H~07FFH;第2组,0800H~0FFFH;第3组,1000H~17FFH;第4组,1800H~1FFFH。地址0B1FH处于第2组内,其芯片的最小地址为0800H。

*【例5.28】 下列有关RAM和ROM的叙述中,正确的是________。

Ⅰ. RAM是易失性存储器,ROM是非易失性存储器

Ⅱ. RAM和ROM都采用随机存取方式进行信息访问

Ⅲ. RAM和ROM都可用作Cache

Ⅳ. RAM和ROM都需要进行刷新

A. 仅Ⅰ和Ⅱ　　B. 仅Ⅱ和Ⅲ　　C. 仅Ⅰ、Ⅱ和Ⅳ　　D. 仅Ⅱ、Ⅲ和Ⅳ

解:A。

分析:RAM中的内容断电后即丢失(易失性),ROM中的内容断电后不会丢失(非易失性),同时RAM和ROM都采用随机存取方式(即CPU对任何一个存储单元的存取时间相同),区别在于RAM可读可写,ROM只读不写。ROM显然不可用作Cache,也不需要刷新。

*【例5.29】 下列命中组合情况中,一次访存过程中不可能发生的是________。

A. TLB未命中,Cache未命中,Page未命中

B. TLB未命中,Cache命中,Page命中

C. TLB命中,Cache未命中,Page命中

D. TLB命中,Cache命中,Page未命中

解:D。

分析:TLB(快表)和慢表(页表,Page)构成二级存储系统,若TLB命中,则Page必命中。因此不可能发生的是D选项。

本题看似既涉及虚拟存储器,又涉及Cache存储器,实际上这里并不需要考虑Cache的命中与否。因为一旦页缺失,说明信息不在主存,那么快表(TLB)中就一定没有该页表项,所以不存在TLB命中,Page缺失的情况,根本谈不上访问Cache是否命中。

*【例5.30】 下列各类存储器中,不采用随机存取方式的是________。

A. EPROM　　B. CDROM　　C. DRAM　　D. SRAM

解:B。

分析:CDROM是只读的光盘存储器。在4类存储器中,只有CDROM属于辅助存储器,不能采用随机存取方式。

*【例5.31】 某计算机存储器按字节编址,主存地址空间大小为64MB,现用4M×8的RAM芯片组成32MB的主存储器,则存储器地址寄存器MAR的位数至少是________。

A. 22位　　B. 23位　　C. 25位　　D. 26位

解:D。

分析:虽然实际的主存储器(RAM区)只有32MB,但不排除还有ROM区,考虑到存储器扩展的需要,MAR应保证能访问到整个主存地址空间。因为主存的地址空间大小为64MB,所以MAR的位数至少需要26位。

*【例5.32】 某计算机存储器按字节编址,采用小端方式存放数据。假定编译器规定int和short型长度分别为32位和16位,并且数据按边界对齐存储。某C语言程序段

如下：

```
struct {
        int     a;
        char    b;
        short   c;
       } record;
record.a=273;
```

若 record 变量的首地址为 0xC008，则地址 0xC008 中内容及 record. c 的地址分别________。

A. 0x00、0xC00D　　B. 0x00、0xC00E

C. 0x11、0xC00D　　D. 0x11、0xC00E

解： D。

分析： 32 位整数 a 需要占 4 个字节，16 位整数 c 需要占 2 个字节，而字符数据 b 占一个字节。$a=273=111$H，采用小端方式存放数据，地址 0xC008 中的内容为 11H。由于数据按边界对齐存储，地址 0xC008～0xC00B 中存放 a，地址 0xC00C 中存放 b，地址 0xC00D 中空闲，地址 0xC00E～0xC00F 中存放 c。

*【例 5.33】 下列关于闪存(Flash Memory)的叙述中，错误的是________。

A. 信息可读可写，并且读、写速度一样快

B. 存储元由 MOS 管组成，是一种半导体存储器

C. 掉电后信息不丢失，是一种非易失性存储器

D. 采用随机访问方式，可替代计算机外部存储器

解： A。

分析： 闪速是一种半导体存储器，它既可在不加电的情况下长期保存信息，又能在线进行快速擦除与重写。由于它容量大，常用作外部存储器。根据闪存的特性，采用排除法，很容易得出答案。

*【例 5.34】 假设某计算机按字编址，Cache 有 4 个行，Cache 和主存之间交换的块大小为 1 个字。若 Cache 的内容初始为空，采用 2 路组相联映射方式和 LRU 替换算法，当访问的主存地址依次为 0、4、8、2、0、6、8、6、4、8 时，命中 Cache 的次数是________。

A. 1　　B. 2　　C. 3　　D. 4

解： C。

分析： Cache 有 4 个行，2 路组相联，即 Cache 被分成 2 组，每组 2 行。Cache 初始为空，采用 LRU 替换算法，当访问主存的 10 个地址依次为 0、4、8、2、0、6、8、6、4、8 时，命中 Cache 的次数共有 3 次，分别发生在第 7、8 步和第 10 步时。

注意：此题的结果是按图 5-6(b)方案得出的，如果采用图 5-6(a)方案，则正确的选项应该为 A，即命中 Cache 的次数共有 1 次，发生在第 8 步时，因为访问主存的 10 个地址都是偶地址，只能映射到 Cache 的第 0 组。

*【例 5.35】 某计算机主存地址空间大小为 256MB，按字节编址。虚拟地址空间大小为 4GB，采用页式存储管理，页面大小为 4KB，TLB(快表)采用全相联映射，有 4 个页表项，内容如表 5-7 所示。

表 5-7 例 5.35 页表

有效位	标记	页框号	…
0	FF180H	0002H	…
1	3FFF1H	0035H	…
0	02FF3H	0351H	…
1	03FFFH	0153H	…

则对虚拟地址 03FF F180H 进行虚实地址变换的结果是________。

A. 015 3180H　　B. 003 5180H　　C. TLB 缺失　　D. 缺页

解：A。

分析：虚存地址空间 4GB，则虚地址长度为 32 位，主存地址空间 256MB，则主存地址长度为 28 位。页面大小 4KB，则页内地址长度为 12 位。虚拟地址 03FF F180H 中 180H 为页内地址，03FFFH 为虚页号，查 TLB(快表)发现，该页在主存中，其实页号为 0153H，所以虚实地址变换后的结果为 015 3180H。

*【例 5.36】 某容量为 256MB 的存储器由若干 4M×8 的 DRAM 芯片构成，该 DRAM 芯片的地址引脚和数据引脚总数是________。

A. 19　　B. 22　　C. 30　　D. 36

解：A。

分析：DRAM 有行地址线和列地址线，复用地址引脚。因为 4M×8 的芯片需要 22 位地址，行/列地址各 11 位，故地址引脚 11 条，数据引脚 8 条，合计 19 条。此题容易误选 C，是因为没有考虑行/列地址线的复用。

*【例 5.37】 采用指令 Cache 与数据 Cache 分离的主要目的是________。

A. 降低 Cache 的缺失损失　　B. 提高 Cache 的命中率

C. 降低 CPU 平均访存时间　　D. 减少指令流水线资源冲突

解：D。

分析：采用指令 Cache 与数据 Cache 分离最根本的原因是避免取指令和取数据时发生冲突，以减少指令流水线资源冲突。

*【例 5.38】 某计算机的主存地址空间大小为 256MB，按字节编址，指令 Cache 和数据 Cache 分离，均有 8 个 Cache 行，每个 Cache 行大小为 64B，数据 Cache 采用直接映射方式。现有两个功能相同的程序 A 和 B，其伪代码如下：

程序 A:

```
int a[256][256];
 ⋮
int sum_array1()
{
    int i,j,sum=0
    for(i=0,i<256;i++)
        for(j= 0;j<256;j++)
            sum+=a[i][j];
    return sum;
}
```

程序 B:

```
int a[256][256];
 ⋮
int sum_array2()
{
    int i,j,sum=0
    for(j=0,j<256;j++)
        for(i= 0;i<256;i++)
            sum+=a[i][j];
    return sum;
}
```

假定 int 类型数据用 32 位补码表示，程序编译时 i、j、sum 均分配在寄存器中，数组 *a* 按行优先方式存放，首地址 320（十进制数）。请回答下列问题，要求说明理由或给出计算过程。

（1）若不考虑用于 Cache 一致性维护和替换算法的控制位，则数据 Cache 的总容量为多少？

（2）数组数据 *a*[0][31]和 *a*[1][1]各自所在的主存块对应的 Cache 行号分别是多少（Cache 行号从 0 开始）？

（3）程序 A 和 B 的数据访问命中率各是多少？哪个程序的执行时间更短？

解：（1）数据 Cache 有 8 个 Cache 行，每个 Cache 行大小为 64B。若不考虑用于 Cache 一致性维护和替换算法的控制位，则数据 Cache 的总容量为 8×64B=512B。

（2）数据 Cache 容量为 512B，Cache 地址为 9 位，有 8 个 Cache 行，块号 3 位，块的大小为 64B，块内地址 6 位。主存容量为 256MB，按字节编址，256MB=2^{28}B，主存地址 28 位，其中块标记为 19 位，块号 3 位，块内地址 6 位。主存和 Cache 的地址格式如图 5-22 所示。

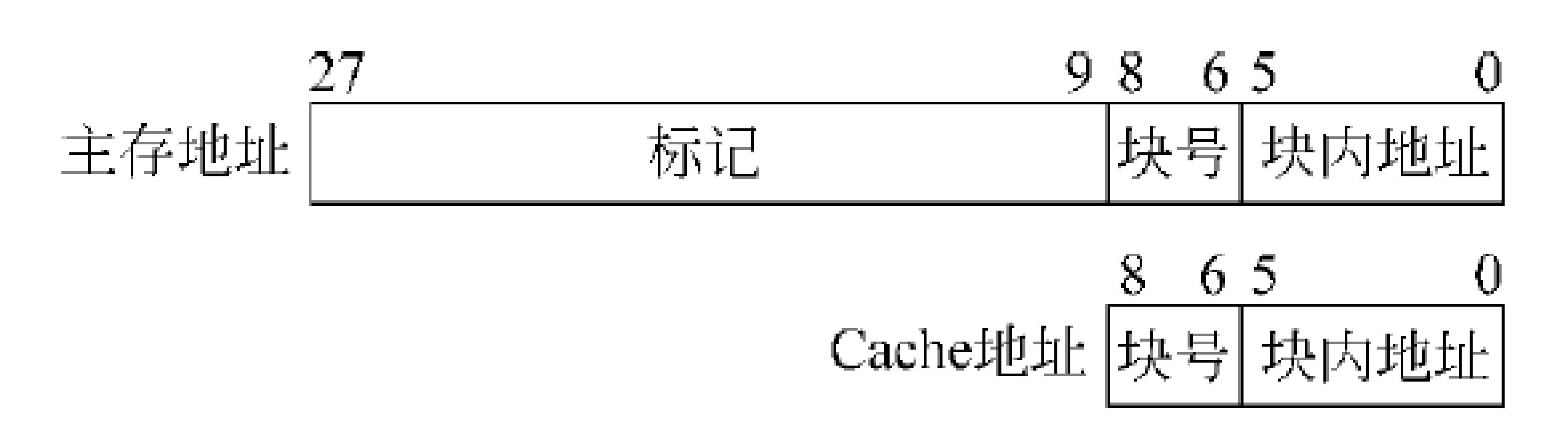


图 5-22　例 5.38 主存和 Cache 的地址格式

数组按行优先方式存放，首地址 320，数组元素占 4 个字节。数据 Cache 采用直接映射方式。*a*[0][31]的地址为 320+31×4=444=110111100B，主存块 110 对应的 Cache 行号为 110B=6；*a*[1][1] 的地址为 320+(256+1)×4=1348=10101000100B，主存块 10101 对应的 Cache 行号为 101B=5。

（3）数组 *a* 存放的数据量为 256×256×4B=2^{18}B，占用 2^{18}÷64=2^{12}个主存块，按行优先存放，程序 A 逐行访问数组 *a*，共需要访存的次数为 2^{16}次，未命中次数 2^{12}，于是程序 A 的数据访问命中率为：$(2^{16}-2^{12})\div 2^{16}\times 100\%=93.75\%$。

程序 B 逐列访问数组 *a*，由于数组 *a* 一行的数据量为 1KB>64B，所以访问第 0 列每个元素均不命中。由于数组有 256 列，数据 Cache 仅有 8 行，故访问数组后续列元素时仍然不命中，于是程序 B 的数据访问命中率为 0%。

程序 A 与程序 B 的区别在于是行优先遍历还是列优先遍历，而数组 *a* 是按行优先方式存放的，所以行优先遍历比列优先遍历命中率高得多，由于从 Cache 读数据比从主存读数据快得多，所以程序 A 的执行过程更短。

分析：本题涉及程序访问的局部性，程序的局部性包括：时间局部性和空间局部性。数组数据在主存中连续存放，为了更好地利用程序访问的空间局部性，通常把当前访问单元以及邻近单元作为一个主存块一起调入 Cache，这个主存块的大小以及程序对数组元素的访问顺序等都对程序的性能有一定的影响。程序 A 和程序 B 的区别在于，程序 A 对数组 *a* 的访问次序是 *a*[0][0]、*a*[0][1]，…，*a*[0][255]；*a*[1][0]，…，*a*[1][255]；…，*a*[255][255]，访问顺序和存放顺序一致，空间局部性好；而程序 B 对数组 *a* 的访问次序是 *a*[0][0]，*a*[1][0]，…，*a*[255][0]；*a*[0][1]，…，*a*[255][1]；…，*a*[255][255]，访问顺序和存放不顺序

一致,每次访问都要跳过256个单元,即每次装入一个主存块到Cache时,下一个要访问的数组元素都不能装入Cache,因而没有空间局部性。

*【例5.39】 某计算机按字节编址,虚拟(逻辑)地址空间大小为16MB,主存(物理)地址空间大小为1MB,页面大小为4KB;Cache采用直接映射方式,共8行;主存与Cache之间交换的块大小为32B。系统运行到某一时刻时,页表的部分内容如图5-23所示,Cache的部分内容如图5-24所示,图中页框号及标记字段的内容为十六进制形式。

虚页号	有效位	页框号	…
0	1	06	…
1	1	04	…
2	1	15	…
3	1	02	…
4	0	—	…
5	1	2B	…
6	0	—	…
7	1	32	…

图5-23 例5.39页表的部分内容

行号	有效位	标记	…
0	1	020	…
1	0	—	…
2	1	01D	…
3	1	105	…
4	1	064	…
5	1	14D	…
6	0	—	…
7	1	27A	…

图5-24 例5.39Cache的部分内容

请回答下列问题。

(1) 虚拟地址共有几位,哪几位表示虚页号?物理地址共有几位,哪几位表示页框号(物理页号)?

(2) 使用物理地址访问Cache时,物理地址应划分哪几个字段?要求说明每个字段的位数及在物理地址中的位置。

(3) 虚拟地址001C60H所在的页面是否在主存中?若在主存中,则该虚拟地址对应的物理地址是什么?访问该地址时是否Cache命中?要求说明理由。

(4) 假定为该机配置一个4路组相联的TLB,该TLB共可存放8个页表项,若其当前内容(十六进制)如图5-25所示,则此时虚拟地址024BACH所在的页面是否在主存中?要求说明理由。

组号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号
0	0	—	—	1	001	15	0	—	—	1	012	1F
1	1	013	2D	0	—	—	1	008	7E	0	—	—

图5-25 例5.39TLB的部分内容

解:(1) 由于页面大小为4KB,页内地址需要12位,所以虚拟地址24位,其中虚页号占12位;物理地址20位,其中页框号(实页号)占8位。

(2) 主存物理地址20位,从左至右应划分3个字段:标记字段、块(行)号字段、块内地址字段。其中标记12位,块(行)号3位,块内地址5位。

(3) 虚拟地址001C60H=0000 0000 0001 1100 0110 0000B,该虚拟地址的虚页号为001H,查页表可以发现,虚页号1对应的有效位为"1",表明此页在主存中,页框号为04H,对应的20位物理地址是04C60H=0000 0100 1100 0110 0000。访问该地址时,Cache不命中,因为Cache采用直接映射方式,对应的物理地址应该映射到Cache的第3行中,其有效

位为1,标记(值为105H)≠04CH(物理地址高12位),故访问该地址时Cache不命中。

(4) 虚拟地址024BACH=0000 0010 0100 1011 1010 1100B,虚页号为024H,TLB中存放8个页表项,采用4路组相联,即TLB分为2组,每组4个页表项。12位虚页号字段中最低位作为组索引,其余11位为标记位。现在最低位为0,表明选择第0组,11位的标记为012H,根据标记可以知道TLB命中,所在的页面在主存中。因为如果在TLB中查到了页表项,即TLB命中,说明所在页一定命中。

分析: 本题涉及主存、Cache和虚拟存储器。根据题目中给出的条件:虚存为16MB、主存为1MB、页面大小为4KB、Cache中块大小32B可知,虚拟地址24位,主存物理地址20位,Cache地址8位(其中块内地址5位)。

第(1)和(2)小题没有什么难度,很容易得出结果,但(3)和(4)题有一定难度,需要仔细分析,要求掌握从虚拟地址转换到物理地址直至产生Cache地址的过程。

*【例5.40】 假定某计算机的CPU主频为80 MHz,CPI为4,并且平均每条指令访存1.5次,主存与Cache之间交换的块大小为16 B,Cache的命中率为99%,存储器总线宽度为32位。请回答下列问题。

(1) 该计算机的MIPS数是多少?平均每秒Cache缺失的次数是多少?在不考虑DMA传送的情况下,主存带宽至少达到多少才能满足CPU的访存要求?

(2) 假定在Cache缺失的情况下访问主存时,存在0.0005%的缺页率,则CPU平均每秒产生多少次缺页异常?若页面大小为4KB,每次缺页都需要访问磁盘,访问磁盘时DMA传送采用周期挪用方式,磁盘I/O接口的数据缓冲寄存器为32位,则磁盘I/O接口平均每秒发出的DMA请求次数至少是多少?

(3) CPU和DMA控制器同时要求使用存储器总线时,哪个优先级更高?为什么?

(4) 为了提高性能,主存采用4体交叉存储模式,工作时每1/4个存储周期启动一个体。若每个体的存储周期为50ns,则该主存能提供的最大带宽是多少?

解: (1) 平均每秒CPU执行的指令数$=\dfrac{\text{主频}}{\text{CPI}}=80\text{MHz}\div 4=20\text{MIPS}=20\times 10^6$。

平均每秒Cache缺失的次数为$20\times 10^6\times 1.5\times(1-99\%)=300\ 000$。

当Cache缺失时,CPU访问主存,主存与Cache之间以块为单位传送数据,此时主存带宽为16B×300 000/s=4.8MB/s。在不考虑DMA传输的情况下,主存带宽至少达到4.8MB/s才能满足CPU的访存要求。

(2) 平均每秒钟“缺页”异常次数为300 000×0.0005%=1.5次。

因为存储器总线宽度为32位,所以,每传送32位数据,磁盘控制器发出一次DMA请求,故平均每秒磁盘DMA请求的次数至少为1.5×4KB÷4B=1.5K=1536。

(3) CPU和DMA控制器同时要求使用存储器总线时,DMA请求优先级更高;因为若DMA请求得不到及时响应,I/O传输数据可能会丢失。

(4) 4体交叉存储模式能提供的最大带宽为4×4B÷50ns=320MB/s。

分析: 本题是一道涉及多个知识点的综合题。如第(1)小题中涉及计算机的性能指标,求解运算速度。其他几个小题涉及Cache缺失(不命中)时CPU访问主存、访问主存缺页时访问磁盘,以及访问磁盘时采用DMA传送的问题,还涉及主存带宽和多体交叉存储器的问题。

*【例 5.41】 某 32 位计算机,CPU 主频为 800MHz,Cache 命中时的 CPI 为 4,Cache 块大小为 32 字节;主存采用 8 体交叉存储方式,每个体的存储字长为 32 位、存储周期为 40ns;存储器总线宽度为 32 位,总线时钟频率为 200MHz,支持突发传送总线事务。每次读突发总线事务的过程包括:送首地址和命令、存储器准备数据、传送数据。每次突发传送 32 字节,传送地址或 32 位数据均需要一个总线时钟周期。请回答下列问题,要求给出理由或计算过程。

(1) CPU 和总线的时钟周期各为多少?总线的带宽(即最大数据传输率)为多少?

(2) Cache 缺失时,需要用几个读突发传送总线事务来完成一个主存块的读取?

(3) 存储器总线完成一次读突发传送总线事务所需的时间是多少?

(4) 若程序 BP 执行过程中,共执行了 100 条指令,平均每条指令需进行 1.2 次访存,Cache 的缺失率为 5%,不考虑替换等开销,则 BP 的 CPU 执行时间是多少?

解:(1) CPU 的时钟周期为 1÷800MHz=1.25ns。

总线的时钟周期为 1÷200MHz=5ns。

总线带宽为 4B×200MHz=800MB/s 或 4B÷5ns=800MB/s

(2) 由于每次突发传送 32 个字节,而 Cache 块的大小正好也为 32 个字节,所以 Cache 缺失时需要用一个读突发传送总线事务来完成一个主存块的读取。

(3) 一次读突发传送总线事务包括一次地址传送和 32 个字节的数据传送:用一个总线时钟周期传输地址(5ns),每隔 40ns÷8=5ns 启动一个体工作,第一个体读取数据花费 40ns,之后的数据存取和数据传输时间重叠,共需用 8 个总线时钟周期传输数据。故读突发传送总线事务时间为 5ns+40ns+8×5ns=85ns。

(4) 程序 BP 的 CPU 执行时间包括 Cache 命中时的执行时间和 Cache 缺失时带来的额外开销两部分。Cache 命中时的指令执行时间为:指令条数×CPI×时钟周期=100×4×1.25ns=500ns。指令执行过程中 Cache 缺失时的额外开销为:指令条数×访存次数×缺失率×读突发传送总线事务时间=1.2×100×5%×85ns=510ns。所以程序 BP 的执行时间为 500ns+510ns=1010ns。

分析:本题是一道涉及多个知识点的综合题。总线事务是指从请求总线到完成总线使用的操作序列,它是在一个总线周期中发生的一系列活动。突发传送又称猝发传送,是一种总线传输方式,即在一个总线周期中可以传输多个存储地址连续的数据。

*【例 5.42】 假设对于例 3.25 中的计算机 M 和程序段 P 的机器代码,M 采用页式虚拟存储管理;P 开始执行时,(R1)=(R2)=0,(R6)=1000,其机器代码已调入主存但不在 Cache 中,数组 *A* 未调入主存,且所有数组元素在同一页,并存储在磁盘同一个扇区。请回答下列问题,并说明理由。

(1) P 执行结束后,R2 的内容是多少?

(2) M 的指令 Cache 和数据 Cache 分离。若指令 Cache 共有 16 行,Cache 和主存交换的块大小为 32 字节,则其数据区的容量是多少?若仅考虑程序段 P 的执行,则指令 Cache 的命中率是多少?

(3) P 在执行过程中,哪条指令的执行可能发生溢出异常?哪条指令的执行可能发生缺页异常?对于数组 *A* 的访问,需要读磁盘和 TLB 至少各多少次?

解:由于在例 3.25 已经详细分析了程序段 P 的各条指令,在此重点讨论存储管理的

问题。

(1) 程序 P 执行结束后，R2 的内容为 1000。因为这是一段循环程序，R2 初始值为 0，每循环一次，(R2)+1，当(R2)=(R6)时，结束循环。

(2) 指令 Cache 数据区的容量：16×32B=512B，指令 Cache 的命中率为 99.98%。由于程序 P 共有 6 条指令，每条指令占 4B，共 24B，小于 Cache 和主存交换的块大小(32B)，且其起始地址为 08048100H，因而所有指令都在同一个主存块中。因为程序 P 已调入主存但不在 Cache 中，读取第一条指令时，发生 Cache 缺失，故将 P 所在主存块调入 Cache 某一行，以后每次读取指令时，都能在指令 Cache 中命中。因此，P 在 1000 次循环执行过程中，共发生 1 次指令访问缺失，故指令 Cache 的命中率为(1000×6−1)÷(1000×6)=99.98%。

(3) 程序 P 执行过程中，指令 4(add R1,R1,R5)的执行可能发生溢出异常，因为只有这一条指令是真正的求和指令，sum 有可能发生溢出。

指令 3(load R5,0(R4))的执行可能会发生缺页异常，因为 load 指令需要读取数组 *A* 的内容，当数组 *A* 不在主存中，发生缺页异常。

对于数组 *A* 的访问，需要至少读磁盘 1 次、至少读 TLB 1001 次。因为第一次执行 load 指令时，数组 *A* 未调入主存，故访问 TLB 缺失，并发生缺页，需要从磁盘上读取数组 *A*，而数组 *A* 所在页在同一个磁盘扇区中，所以在不考虑页面置换的情况下，只要读取磁盘 1 次。缺页异常处理结束后，重新执行 load 指令，load 指令的随后 1000 次执行中，每次都能在 TLB 中命中，因而无须访问内存页表项和磁盘，所以 P 在 1000 次循环执行过程中，对于数组 *A*，需要读取 TLB 共 1001 次。

分析：本题是一道与例 3.25 直接相关联的涉及多个知识点的综合题。需要对高级语言、汇编语言的关系很清楚，对从 Cache、主存、磁盘三级存储系统中读取指令和数据的过程很清晰。其中，第(1)小题没有难度，第(2)小题难度也不高，但第(3)小题难度较大。

5.4 同步测试习题及解答

5.4.1 同步测试习题

一、填空题

1. 在多级存储体系中，Cache 的主要功能是________，虚拟存储器的主要功能是________。

2. SRAM 靠________存储信息，DRAM 靠________存储信息。________存储器需要定时刷新。

3. 动态半导体存储器的刷新一般有________、________和________。

4. 一个 512KB 的存储器，其地址线和数据线的总和是________。

5. 若 RAM 芯片内有 1024 个单元，用单译码方式，地址译码器有________条输出线；用双译码方式，地址译码器有________条输出线。

6. 高速缓冲存储器中保存的信息是主存信息的________。

二、选择题

1. 在磁盘和磁带这两种磁介质存储器中，存取时间与存储单元的物理位置有关，按存

储方式分________。

A. 二者都是顺序存取　　B. 二者都是直接存取

C. 磁盘是直接存取,磁带是顺序存取　　D. 磁带是直接存取,磁盘是顺序存取

2. 存储器进行一次完整的读写操作所需的全部时间称为________。

A. 存取时间　　B. 存取周期　　C. CPU 周期　　D. 机器周期

3. 若存储周期 250ns ,每次读出 16 位,则该存储器的数据传送率为________。

A. 4×10^6B/s　　B. 4MB/s　　C. 8×10^6B/s　　D. 8MB/s

4. 用户程序所放的主存空间属于________。

A. 随机存取存储器　　B. 只读存储器

C. 顺序存取存储器　　D. 直接存取存储器

5. 以下哪种类型的存储器速度最快________。

A. DRAM　　B. ROM　　C. EPROM　　D. SRAM

6. 下述说法中正确的是________。

A. 半导体 RAM 信息可读可写,且断电后仍能保持记忆

B. 动态 RAM 是易失性 RAM,而静态 RAM 中的存储信息是不易失的

C. 半导体 RAM 是易失性 RAM,但只要电源不断电,所存信息是不丢失的

D. 半导体 RAM 是非易失性的 RAM

7. 若数据在存储器中采用以低字节地址为字地址的存放方式,则十六进制数 12345678H 的存储字节顺序按地址由小到大依次为________。

A. 12345678　　B. 78563412　　C. 87654321　　D. 34127856

8. 在对破坏性读出的存储器进行读写操作时,为维持原存信息不变,必须辅以的操作是________。

A. 刷新　　B. 再生　　C. 写保护　　D. 主存校验

9. 动态 RAM 的刷新是以________为单位进行的。

A. 存储单元　　B. 行　　C. 列　　D. 存储位

10. SRAM 芯片,其容量为 1024×8,除电源和接地端外,该芯片最少引出线数为________。

A. 16　　B. 17　　C. 20　　D. 21

11. 存储器容量为 $32K\times16$,则________。

A. 地址线为 16 根,数据线为 32 根　　B. 地址线为 32 根,数据线为 16 根

C. 地址线为 15 根,数据线为 16 根　　D. 地址线为 16 根,数据线为 15 根

12. 某计算机字长为 32 位,存储器容量为 4MB,若按字编址,其寻址范围是 0 到________。

A. $2^{20}-1$　　B. $2^{21}-1$　　C. $2^{23}-1$　　D. $2^{24}-1$

13. 设机器字长为 32 位,一个容量为 16MB 的存储器,CPU 按半字寻址,其可寻址的单元数是________。

A. 2^{24}　　B. 2^{23}　　C. 2^{22}　　D. 2^{21}

14. 下述说法正确的是________。

A. EPROM 是可改写的,因而也是随机存储器的一种

B. EPROM 是可改写的，但它不能用作为随机存储器用

C. EPROM 只能改写一次，故不能作为随机存储器用

D. EPROM 是只能改写一次的只读存储器

15. 通常计算机的主存储器可采用________。

A. RAM 和 ROM　　B. ROM

C. RAM　　D. RAM 或 ROM

16. 存储器采用部分译码法片选时________。

A. 不需要地址译码器　　B. 不能充分利用存储器空间

C. 会产生地址重叠　　D. CPU 的地址线全参与译码

17. 双端口存储器发生读/写冲突的情况是________。

A. 左端口与右端口的地址码不同　　B. 左端口与右端口的地址码相同

C. 左端口与右端口的数据码相同　　D. 左端口与右端口的数据码不同

18. 如果一个存储单元被访问，则可能这个存储单元会很快地再次被访问，这称为________。

A. 时间局部性　　B. 空间局部性　　C. 程序局部性　　D. 数据局部性

19. 在主存和 CPU 之间增加高速缓冲存储器的目的是________。

A. 解决 CPU 和主存之间的速度匹配问题

B. 扩大主存容量

C. 扩大 CPU 通用寄存器的数目

D. 既扩大主存容量又扩大 CPU 中通用寄存器的数量

20. 在程序的执行过程中，Cache 与主存的地址映射是由________。

A. 操作系统来管理的　　B. 程序员调度的

C. 由硬件自动完成的　　D. 由软、硬件共同完成的

21. 容量为 64 块的 Cache 采用组相联映射方式，字块大小为 128 个字，每 4 块为一组。若主存 4 096 块，且以字编址，那么主存地址和主存标记的位数分别为________。

A. 16，6　　B. 17，6　　C. 18，8　　D. 19，8

22. 采用虚拟存储器的目的是________。

A. 提高主存的速度　　B. 扩大辅存的存取空间

C. 扩大主存的寻址空间　　D. 扩大存储器的寻址空间

23. 下列关于虚拟存储器的论述中，正确的是________。

A. 对应用程序员透明，对系统程序员不透明

B. 对应用程序员不透明，对系统程序员透明

C. 对应用程序员、系统程序员都不透明

D. 对应用程序员、系统程序员都透明

24. 在虚拟存储器中，辅存的编址方式是________。

A. 按信息块编址　　B. 按字编址

C. 按字节编址　　D. 按位编址

25. 虚拟存储器中的页表有快表和慢表之分，下面关于页表的叙述中正确的是________。

A. 快表与慢表都存储在主存中,但快表比慢表容量小

B. 快表采用优化的搜索算法,因此查找速度快

C. 快表比慢表的命中率高,因此快表可以得到更多的搜索结果

D. 快表采用快速存储器件组成,按照查找内容访问,因此比慢表查找速度快

三、判断题

1. 存取周期是指启动一次存储器操作到完成该操作所需的时间。 (　　)

2. CPU访问主存储器的时间是由存储体的容量决定的,存储容量越大,访问存储器所需时间就越长。 (　　)

3. 随机存储器需要定时地进行刷新。 (　　)

4. 因为动态存储器是破坏性读出,所以必须不断地刷新。 (　　)

5. 断电后,RAM中的数据不会丢失。 (　　)

6. 集中刷新方式在刷新时间内并不影响CPU的读写操作。 (　　)

7. 动态RAM的异步刷新方式没有读写死区。 (　　)

8. 断电后,EEPROM中的数据不会丢失。 (　　)

9. 用1024×1芯片构成8KB存储器,CPU提供地址线A_0～A_{15},其中A_0为高位,则加到各芯片地址端的地址线是A_0～A_9。 (　　)

10. 用1024×1芯片组成8KB存储器,地址线A_{15}(高)～A_0(低),应由A_{15}～A_{13}3位地址经译码产生片选信号。 (　　)

11. 一般情况下,ROM和RAM在存储体中是统一编址的。 (　　)

12. 用户编程的地址称为虚地址,通常虚地址的范围要比实地址大得多。 (　　)

四、简答题

1. 说明SRAM的组成结构,与SRAM相比,DRAM在电路组成上有什么不同之处?

2. DRAM存储器为什么要刷新?采用何种方式刷新?

3. 存储器系统的层次结构可以解决什么问题?实现存储器层次结构的先决条件是什么?用什么来度量?

五、分析题

1. 某计算机系统字长32位,主存以字节编址,试画出存储器字地址和字节地址的分配情况示意图。

2. 某存储器容量为4KB。其中,ROM 2KB,选用EPROM 2K×8;RAM 2KB,选用RAM 1K×8;地址线A_{15}～A_0。写出全部片选信号的逻辑式。

3. 要求用128K×16的SRAM芯片组成512K×16的随机存储器,用64K×16的EPROM芯片组成128K×16的只读存储器。试问:

(1) 数据寄存器多少位?

(2) 地址寄存器多少位?

(3) 两种芯片各需多少片?

(4) 若EPROM的地址从00000H开始,RAM的地址从60000H开始,写出各芯片的地址分配情况。

4. 已知地址总线A_{15}～A_0,其中A_0是最低位。用ROM芯片(4K×4)和RAM芯片(2K×8)组成一个半导体存储器,按字节编址。该存储器ROM区的容量为16KB,RAM的

容量为10KB。

（1）组成该存储器需用多少块ROM芯片和RAM芯片？

（2）该存储器共需要多少根地址线？ROM芯片、RAM芯片各需连入哪几根地址线？

（3）需设置多少个片选信号，分别写出各片选信号的逻辑式。

5. CPU执行一段程序时，Cache完成存取的次数为1900次，主存完成存取的次数为100次，已知Cache存取周期为50ns，主存存取周期为250ns。求：Cache主存系统的命中率、平均访问时间和效率。

6. 在虚拟地址和物理地址均为32位、页大小为4KB的某种体系结构中，假定存在如表5-8所示的地址映像关系，问：对应于下列虚拟地址的物理地址分别是什么？

（1）22433007H。

（2）13385ABCH。

（3）ABC89011。

表5-8　地址映像

虚　页　号	实　页　号
ABC89H	97887H
13385H	99910H
22433H	00001H
54483H	1A8C2H

六、设计题

1. 某机CPU可提供16条地址线，8条数据线，1条控制线（$R/\overline{W}$），$R/\overline{W}=1$表示读，$R/\overline{W}=0$表示写。现用存储器容量为8KB。拟采用2K×4的芯片。

（1）画出CPU与RAM之间的连接图（译码器自定）。

（2）说明该RAM的地址范围。

2. 某机CPU可寻址的最大存储空间为64KB，存储器按字节编址，CPU的数据总线宽度为8位，可提供一个控制器信号为$\overline{RD}$。目前系统中使用的存储器容量为8KB，其中4KB为ROM。拟采用容量为2K×8的ROM芯片，其地址范围为0000H～0FFFH。4KB为RAM，拟采用4K×2的RAM芯片，其地址范围为4000H～4FFFH。

（1）需RAM和ROM芯片各多少片？

（2）画出CPU与存储器之间的连接图（译码器自定）。

3. 某机CPU可输出数据线8条（$D_7 \sim D_0$），地址线20条（$A_{19} \sim A_0$），控制线1条（$\overline{WE}$）。目前使用的存储空间为48KB，其中：16KB为ROM，拟用8K×8位的ROM芯片；32KB为RAM，拟用16K×4的RAM芯片。

（1）需要两种芯片各多少片？

（2）画出CPU与存储器之间的连线图（译码器自定）。

（3）写出ROM和RAM的地址范围。

4. 某微机的寻址范围为64KB，其存储器选择器信号为M，接有8片8KB的存储器，试回答下列问题：

（1）画出选片译码逻辑图。

（2）写出每片RAM的寻址范围。

（3）如果运行时发现不论往哪片存储器存放8KB数据，以A000H起始地址的存储芯片都有相同的数据，分析故障原因。

（4）若发现译码器中的地址线A_{13}与CPU断线，并搭接到高电平的故障，问后果如何？

5. 设CPU有16根地址线，8根数据线，并用$\overline{MREQ}$作为访存控制信号，用$R/\overline{W}$作为

读写命令信号。自选各类存储芯片，画出CPU与存储芯片的连接图。要求：

(1) 最大8K地址是系统程序区，与其相邻的8K地址是系统程序工作区，最小16K地址是用户程序区。

(2) 写出每片存储芯片的类型及地址范围(用十六进制表示)。

(3) 用一个3-8译码器或其他门电路(门电路自定)详细画出存储芯片的选片逻辑。

6. 利用2716(2K×8)、2114(1K×4)和8205(或74LS138)等集成电路为8位微机设计一个容量为4KB ROM、2KB RAM的存储子系统(ROM安排在主存的低端，RAM紧靠ROM)。要求写出设计步骤。

7. 某半导体存储器容量9K×8，其中ROM区4K×8，可选EPROM芯片2K×8/片。RAM区5K×8，可选SRAM芯片2K×4/片，1K×4/片，地址总线A_{15}～A_0(低)，数据总线D_7～D_0(低)。R/W控制读、写。若有控制信号$\overline{MREQ}$。要求：

(1) 设计并画出该存储器逻辑图。

(2) 注明地址分配与片选逻辑式及片选信号极性。

8. 通常主存储器由RAM和ROM组成，试用图5-26所示的两种芯片(2732和6264)设计一个8位微机系统的主存储器，要求：系统程序区8KB，从0000H地址开始；用户程序区40KB，从4000H地址开始。请指出每种芯片各需要多少块？写出各芯片的地址分配，画出该存储器的逻辑框图(注意地址线、数据线和控制线的连接)。

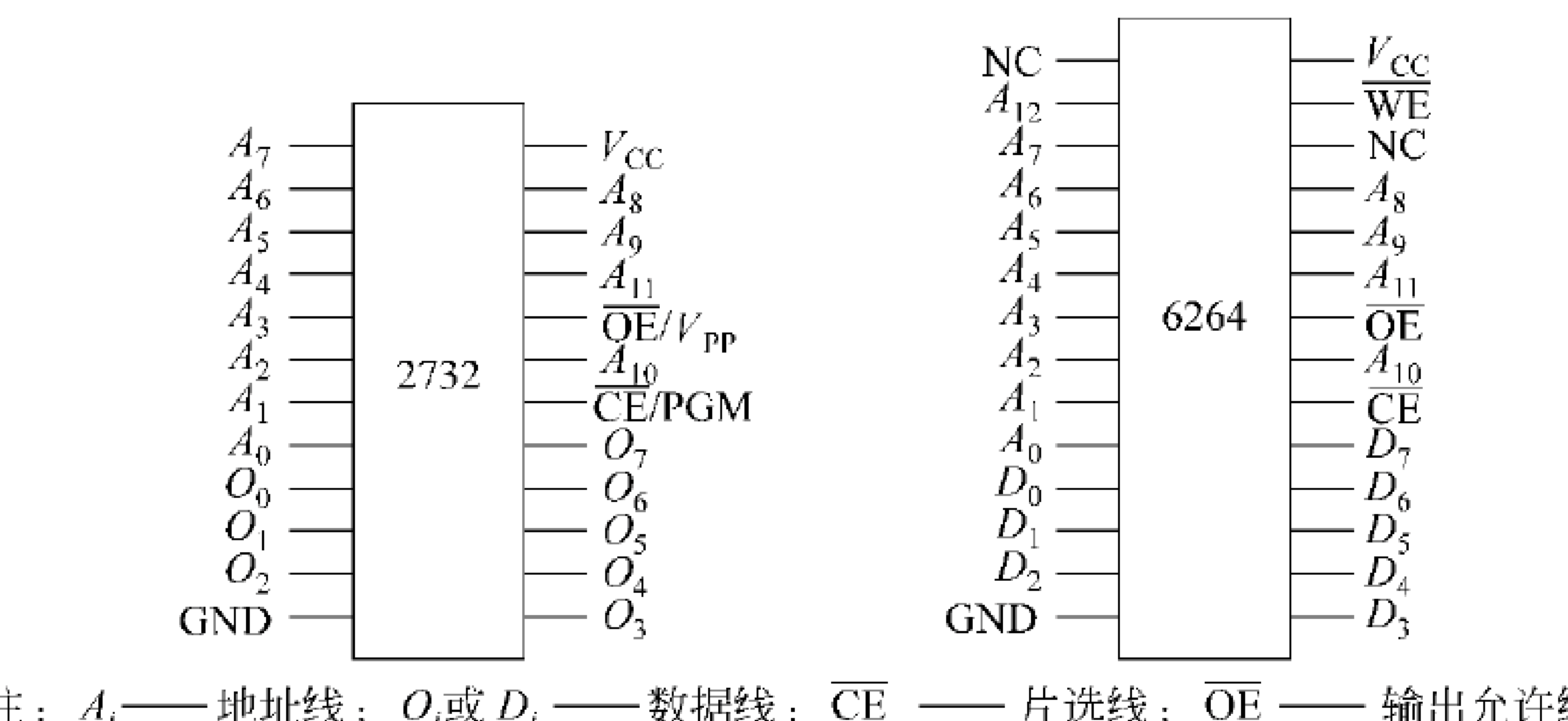


图5-26 使用的芯片

提示：首先根据芯片的管脚图确定出每个芯片的类型(RAM或ROM)和芯片的容量。

9. 假设主存容量为512K×16位，Cache容量为4096×16位，块长为4个16位的字，访存地址为字地址。

(1) 在直接映射方式下，设计主存的地址格式。

(2) 在全相联映射方式下，设计主存的地址格式。

(3) 在二路组相联映射方式下，设计主存的地址格式。

(4) 若主存容量为512K×32位，块长不变，在四路组相联映射方式下，设计主存的地址格式。

5.4.2 同步测试习题解答

一、填空题

1. 提高存储速度，扩大存储容量。

2. 触发器，栅极电容，DRAM。

3. 集中式，分散式，异步式。

4. 27。512KB 的存储器有 19 根地址线，8 根数据线，所以其总和是 27 根。

5. 1024，64。单译码方式只有一个译码器；双译码方式有两个译码器，每个译码器有 32 条输出线。

6. 活跃块的副本。

二、选择题

1. C。磁盘是直接存取存储器，磁带是顺序存取存储器。

2. B。一次完整的读写操作所需的全部时间也就是连续两次访问存储器操作的间隔时间。要注意存取周期与存取时间这两个概念的区别，存取时间是指从启动一次存储器操作到完成该操作所经历的时间，所以存取时间小于存取周期。

3. C。存储周期 $250\text{ns}=250\times10^{-9}\text{s}$，每个存储周期可读出 16 位(2 个字节)，则数据传送率为 $2\text{B}\div(250\times10^{-9})\text{s}=8\times10^{6}\text{B/s}$。此题中选项 D 的问题在于 $1\text{K}\approx10^{3}$，$1\text{M}\approx10^{6}$，有误差。

4. A。用户程序可读可写，存放在 RAM(随机存储器)中。

5. D。SRAM 因为不需要刷新，所以速度最快。

6. C。半导体 RAM，无论静态 RAM 还是动态 RAM 都是易失性的，断电后信息将丢失。

7. B。此存放方式是小端次序方案，将最低有效字节存储在最小地址位置。

8. B。对于破坏性读出的存储器，每当一次读出操作之后，必须紧接一个重写(再生)的操作，以便恢复被破坏的信息，保持原存信息不变。

9. B。动态 RAM 芯片中的全部记忆单元排列成矩阵，刷新是以行为单位进行的，一行中的各记忆单元同时被刷新。

10. C。地址线 10 根，数据线 8 根，控制线(读写和片选)2 根。

11. C。存储器容量为 $32\text{K}\times16$，$32\text{K}=2^{15}$，所以有地址线 15 根，数据线 16 根。

12. A。4MB=1MW。

13. B。$16\text{MB}=2^{24}$，由于字长为 32 位，现在按半字(16 位)寻址，相当于有 8M 个存储单元，每个存储单元中存放 16 位。

14. B。EPROM 是可擦除可改写的，允许改写多次，但它并不是随机存储器，也不能当作随机存储器使用。

15. A。主存由 RAM 和 ROM 组成，其中 ROM 中的信息是不可改变的，RAM 中的信息是可以改变的。

16. C。部分译码即只用高位地址的一部分参与译码，而另一部分高位地址与译码电路无关，所以会出现一个存储单元对应多个地址的现象，这种现象被称为地址重叠。

17. B。双端口存储器设计有两个端口，两套读写逻辑电路。在同时操作同一单元时会

发生冲突,所以地址码相同时产生冲突。

18. A。从时间上看,一个单元刚被访问又被再次访问,这是因为程序中存在着循环。

19. A。增设高速缓冲存储器可以解决CPU和主存之间的速度匹配问题。

20. C。Cache存储系统全部用硬件来调度,对于程序员是透明的。

21. D。主存容量4K×128=512K字,故主存地址19位,由主存标记、组号和块内地址3部分组成。因为字块大小为128个字,故块内地址7位,Cache被分成64÷4=16组,故组号4位,主存标记19−4−7=8位。

22. D。虚拟存储器是为解决主存容量不足而提出来的,以扩大整个存储器的寻址空间。

23. A。由于虚拟存储器需要通过操作系统来调度,因此对系统程序员是不透明的,但对应用程序员是透明的。

24. A。CPU不能直接访问辅存,辅存中的程序和数据在需要时才传送到主存,传送的最小单位是一个信息块。

25. D。快表只是慢表的一个副本,而且只存放了慢表中很少的一部分。快表按内容访问,查表速度快。

三、判断题

1. ×。启动一次存储器操作到完成该操作所需的时间称为存取时间。

2. ×。CPU访问主存储器的时间与存储体的容量无关。

3. ×。随机存储器可分为静态随机存储器和动态随机存储器,只有动态随机存储器需要定时地进行刷新。

4. ×。动态存储器需要刷新的原因不是因为破坏性读出,即使是非破坏性读出的动态存储器也需要定时地刷新。

5. ×。RAM是易失性的存储器,断电后,RAM中的数据会丢失。

6. ×。集中方式在刷新期间,CPU不能访存。

7. ×。异步刷新方式仍然有死区,只是死区比较小而已。

8. √。

9. ×。因为 A_0 是高位,A_{15} 是低位,加到各芯片地址端的是地址线的低10位,为 $A_6 \sim A_{15}$。

10. ×。应由 $A_{12} \sim A_{10}$ 经译码产生片选信号,$A_{15} \sim A_{13}$ 可略去不用(因为总容量只有8KB)。

11. √。

12. √。

四、简答题

1. SRAM由存储体、读写电路、地址译码电路、控制电路组成,DRAM还需要有动态刷新电路。与SRAM相比,DRAM在电路组成上有以下不同之处:

(1) 地址线的引脚一般只有一半,因此,增加了两根控制线 $\overline{RAS}$、$\overline{CAS}$,分别控制接收行地址和列地址。

(2) 没有$\overline{CS}$引脚,在存储器扩展时用$\overline{RAS}$来代替。

2. DRAM 记忆单元是通过栅极电容上存储的电荷来暂存信息的。由于存储的电荷终究会泄漏掉,时间一长,信息就会丢失。为此,必须设法由外界按一定规律给栅极电容充电,这个过程就叫做刷新。

DRAM 是逐行进行刷新的,刷新周期数与 DRAM 的扩展无关,只与单个 DRAM 芯片的内部结构有关,对于一个 128×128 矩阵结构的 DRAM 芯片,只需要 128 个刷新周期。

3. 存储器层次结构可以提高计算机存储系统的性能价格比,即在速度方面接近最高级的存储器,在容量和价格方面接近最低级的存储器。

实现存储器层次结构的先决条件是程序局部性原理,即存储器访问的局部性是实现存储器层次结构的基础。其度量方法主要是存储系统的命中率。

五、分析题

1. 一个字由 4 个字节(B_3、B_2、B_1、B_0)组成,其中 B_3 是字的最高有效字节,B_0 是最低有效字节。图 5-27(a)称为小端方案。假设字地址为 N,则字节 B_3、B_2、B_1、B_0 依次存放在地址为 $N+3$、$N+2$、$N+1$、$N+0$ 的存储单元,即字地址等于最低有效字节地址。图 5-27(b)称为大端方案。假设字地址为 N,则字节 B_3、B_2、B_1、B_0 依次存放在地址为 $N+0$、$N+1$、$N+2$、$N+3$ 的存储单元,即字地址等于最高有效字节地址。

字地址 N	B_3	B_2	B_1	B_0
字节地址	$N+3$	$N+2$	$N+1$	$N+0$

(a)

字地址 N	B_3	B_2	B_1	B_0
字节地址	$N+0$	$N+1$	$N+2$	$N+3$

(b)

图 5-27 字地址和字节地址的分配情况示意图

2. 需要 1 片 EPROM,片内地址 11 位;2 片 RAM,片内地址 10 位。总容量 4KB,需要 12 根地址线,因此有:

EPROM $CS_0=\overline{A_{11}}$

RAM $CS_1=A_{11}\overline{A_{10}}$

$CS_2=A_{11}A_{10}$

3. (1) 数据寄存器 16 位。

(2) 地址寄存器 20 位。

(3) 需 SRAM 芯片 4 片,EPROM 芯片 2 片。

(4) 各芯片的地址分配情况如表 5-9 所示。

4. (1) ROM 芯片:8 片(4 组,每组 2 片),RAM 芯片:5 片(每组 1 片)。

(2) 26KB 共需 15 根地址线(A_{14}~A_0),ROM 芯片连入地址线 A_{11}~A_0,RAM 芯片连入地址线 A_{10}~A_0。

表 5-9　各芯片地址分配

所选芯片	选片地址				片内地址					译码输出	地址分配
	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	…	A_0		
$EPROM_1$	0	0	0	0	0	0	0	…	0	$\overline{Y}_0=0$	00000H～0FFFFH
	0	0	0	0	1	1	1	…	1		
$EPROM_2$	0	0	0	1	0	0	0	…	0	$\overline{Y}_0=0$	10000H～1FFFFH
	0	0	0	1	1	1	1	…	1		
$SRAM_1$	0	1	1	0	0	0	0	…	0	$\overline{Y}_3=0$	60000H～7FFFFH
	0	1	1	1	1	1	1	…	1		
$SRAM_2$	1	0	0	0	0	0	0	…	0	$\overline{Y}_4=0$	80000H～9FFFFH
	1	0	0	1	1	1	1	…	1		
$SRAM_3$	1	0	1	0	0	0	0	…	0	$\overline{Y}_5=0$	A0000H～BFFFFH
	1	0	1	1	1	1	1	…	1		
$SRAM_4$	1	1	0	0	0	0	0	…	0	$\overline{Y}_6=0$	C0000H～DFFFFH
	1	1	0	1	1	1	1	…	1		

(3) 共需要 9 个片选信号。各片选信号的逻辑表达式为：

$$CS_0=\overline{A_{14}}\,\overline{A_{13}}\,\overline{A_{12}}$$
$$CS_1=\overline{A_{14}}\,\overline{A_{13}}A_{12}$$
$$CS_2=\overline{A_{14}}A_{13}\overline{A_{12}}$$
$$CS_3=\overline{A_{14}}A_{13}A_{12}$$
$$CS_4=A_{14}\overline{A_{13}}\,\overline{A_{12}}\,\overline{A_{11}}$$
$$CS_5=A_{14}\overline{A_{13}}\,\overline{A_{12}}A_{11}$$
$$CS_6=A_{14}\overline{A_{13}}A_{12}\overline{A_{11}}$$
$$CS_7=A_{14}\overline{A_{13}}A_{12}A_{11}$$
$$CS_8=A_{14}A_{13}\overline{A_{12}}\,\overline{A_{11}}$$

5. (1) 命中率 $H=\dfrac{1900}{1900+100}=95\%$。

(2) $T_A=H\times T_{A_1}+(1-H)\times T_{A_2}=0.95\times 50\text{ns}+(1-0.95)\times 250\text{ns}=60\text{ns}$。

(3) $e=\dfrac{T_{A_1}}{T_A}=\dfrac{50}{60}=83.3\%$。

6. (1) 对应的物理地址 00001007H。虚拟地址中的低 12 位为页内地址，高 20 位为虚页号。通过查找表 5-8，可以得到对应的实页号。将实页号与页内地址拼接在一起，就得到对应的物理地址。

(2) 对应的物理地址 99910ABCH。

(3) 对应的物理地址 97887011H。

六、设计题

1. (1) 需用芯片数目 $\dfrac{8\text{K}\times 8}{2\text{K}\times 4}=8$(片)，采用字和位同时扩展的方法。将芯片分成 4 组，每组 2 片。选用一个 2-4 译码器产生芯片组的选择信号。连接图略。

(2) 该 RAM 的地址范围为：

第 0 组　0000H～07FFFH

第 1 组　0800H～0FFFH

第 2 组　1000H～17FFH

第 3 组　1800H～1FFFH

2. (1) 需 RAM 芯片 4 片，ROM 芯片 2 片。

(2) 画出 CPU 与存储器之间的连接图如图 5-28 所示。

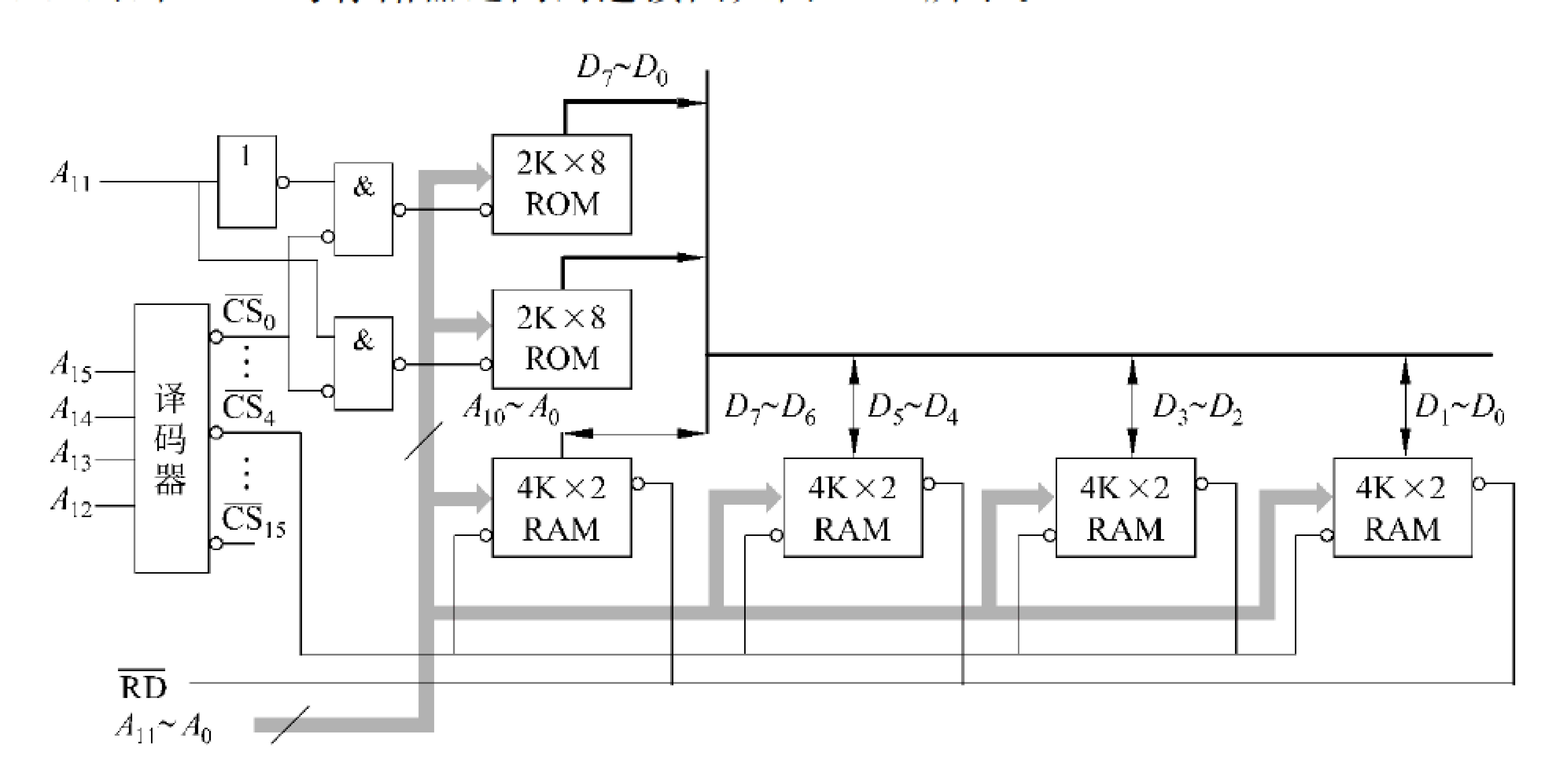


图 5-28　设计题 2 中 CPU 与存储器的连接图

3. (1) 需要 ROM 芯片 2 片，RAM 芯片 4 片。

(2) 画出 CPU 与存储器之间的连接图，如图 5-29 所示。

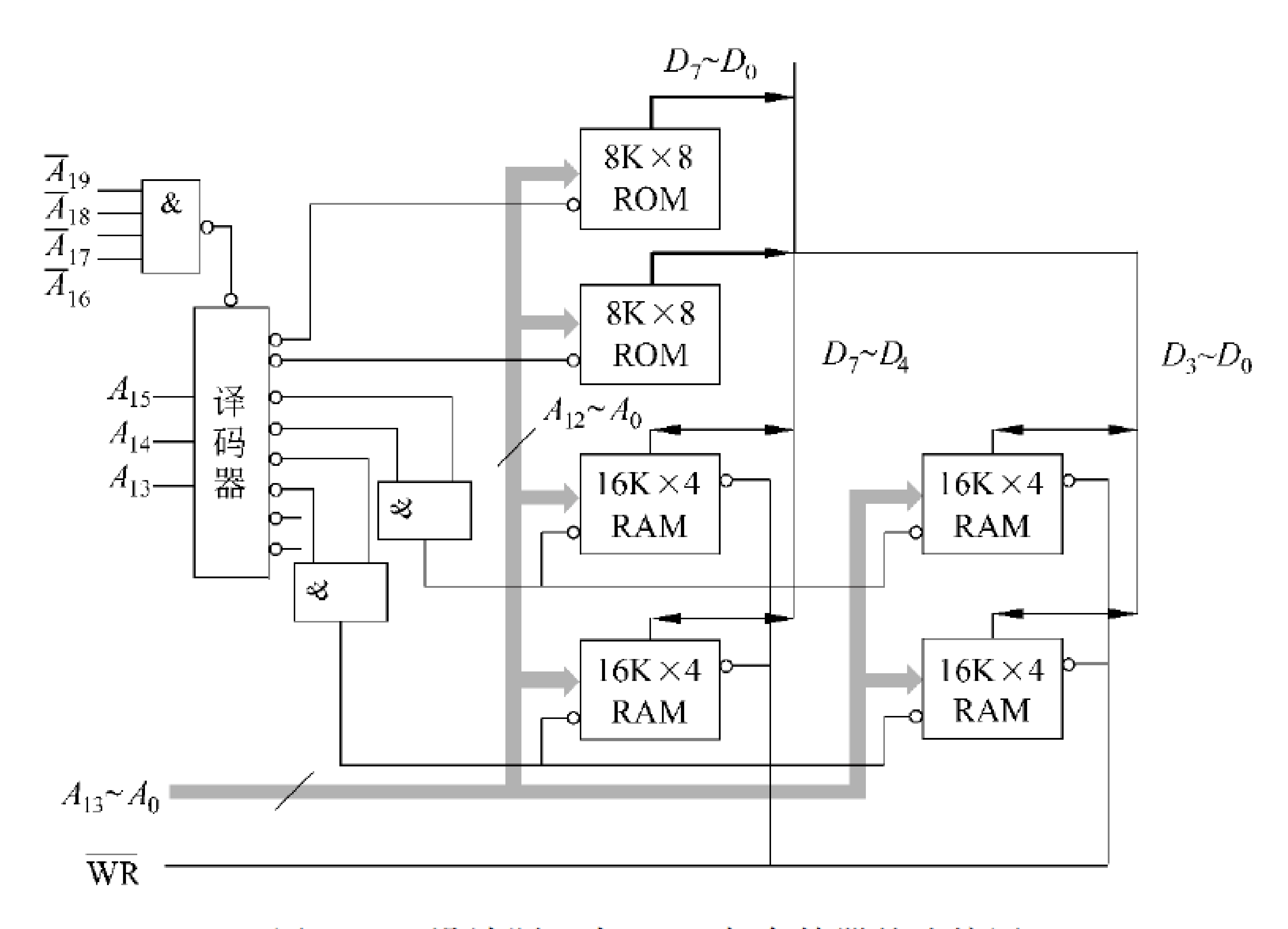


图 5-29　设计题 3 中 CPU 与存储器的连接图

(3) ROM 的地址范围为：

ROM_1　00000H～01FFFH

ROM_2　02000H～03FFFH

RAM 的地址范围为：

RAM_1＋RAM_2　04000H～07FFFH

RAM_3＋RAM_4　08000H～0BFFFH

4.（1）选片译码逻辑如图 5-30 所示。

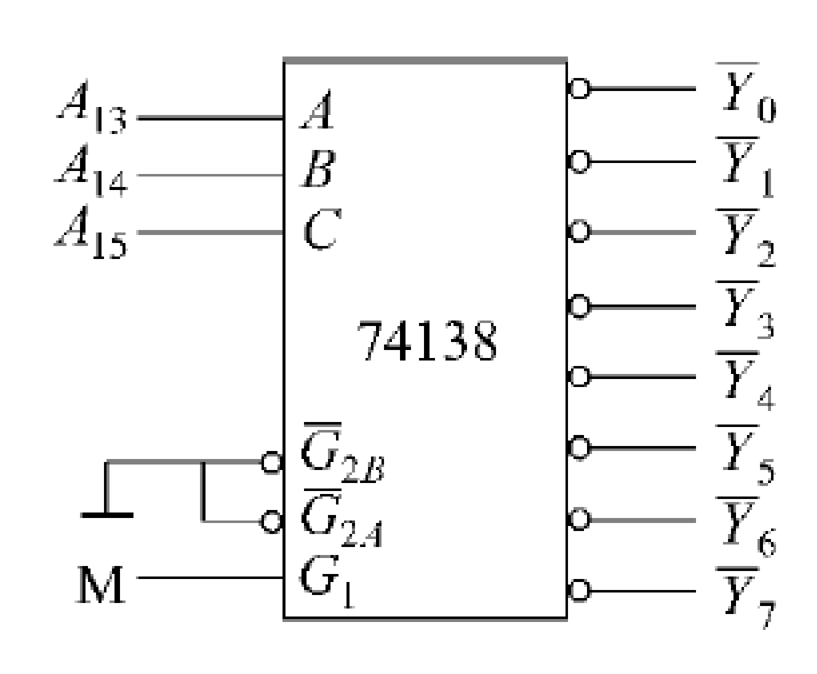


图 5-30　设计题 4 选片译码逻辑

（2）8 片 RAM 的寻址范围分别是：0000H～1FFFH、2000H～3FFFH、4000H～5FFFH、6000H～7FFFH、8000H～9FFFH、A000H～BFFFH、C000H～DFFFH 和 E000H～FFFFH。

（3）说明译码器有误，$\overline{Y_5}$输出始终为低，从而使第 6 片 RAM 始终被选中。

（4）若发现 A_{13}与 CPU 断线，并搭接到高电平的故障，则 $\overline{Y_0}$、$\overline{Y_2}$、$\overline{Y_4}$、$\overline{Y_6}$信号均不会为 0，故第 1、3、5、7 片 RAM 始终不被选中。

5.（1）假设选用 8K×8 的 ROM 和 RAM。系统程序区用 1 片 ROM，系统程序工作区和用户程序区用 3 片 RAM。CPU 与存储芯片的连接图略。

（2）RAM_1(8K×8)0000H～1FFFH 用户程序区

RAM_2(8K×8)2000H～3FFFH 用户程序区

RAM_3(8K×8)C000H～DFFFH 系统程序工作区

ROM(8K×8)E000H～FFFFH 系统程序区

（3）选片逻辑图如图 5-31 所示。

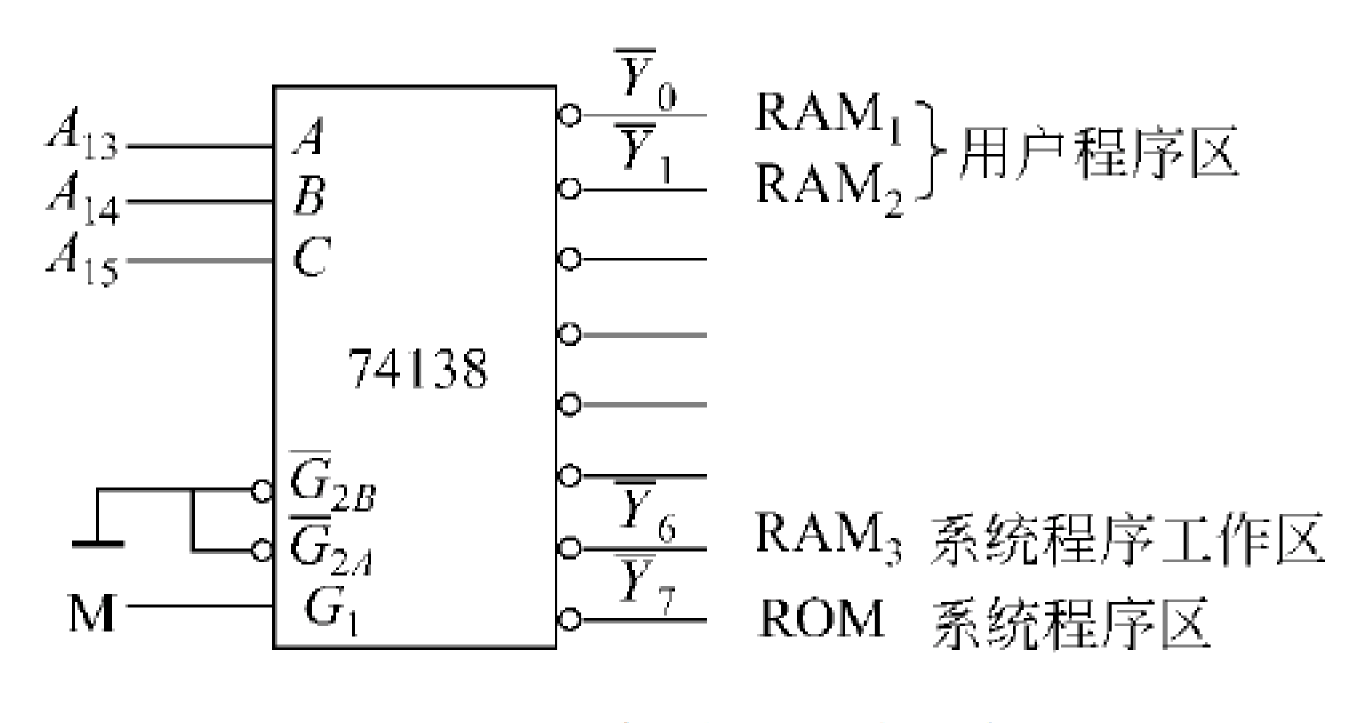


图 5-31　设计题 5 选片逻辑图

6.（1）计算出需要的各种芯片数。需 2716 芯片 2 片，2114 芯片 4 片。

（2）写出每个芯片的地址分配。

ROM_1　0000H～07FFH

ROM_2　0800H～0FFFH

RAM_1＋RAM_2　1000H～13FFH

RAM_3＋RAM_4　1400H～17FFH

（3）设计片选逻辑。

（4）连接存储子系统。

7.（1）存储器逻辑图如图 5-32 所示。

（2）各芯片地址分配。

$EPROM_1$　0000H～07FFH

$EPROM_2$　0800H～0FFFH

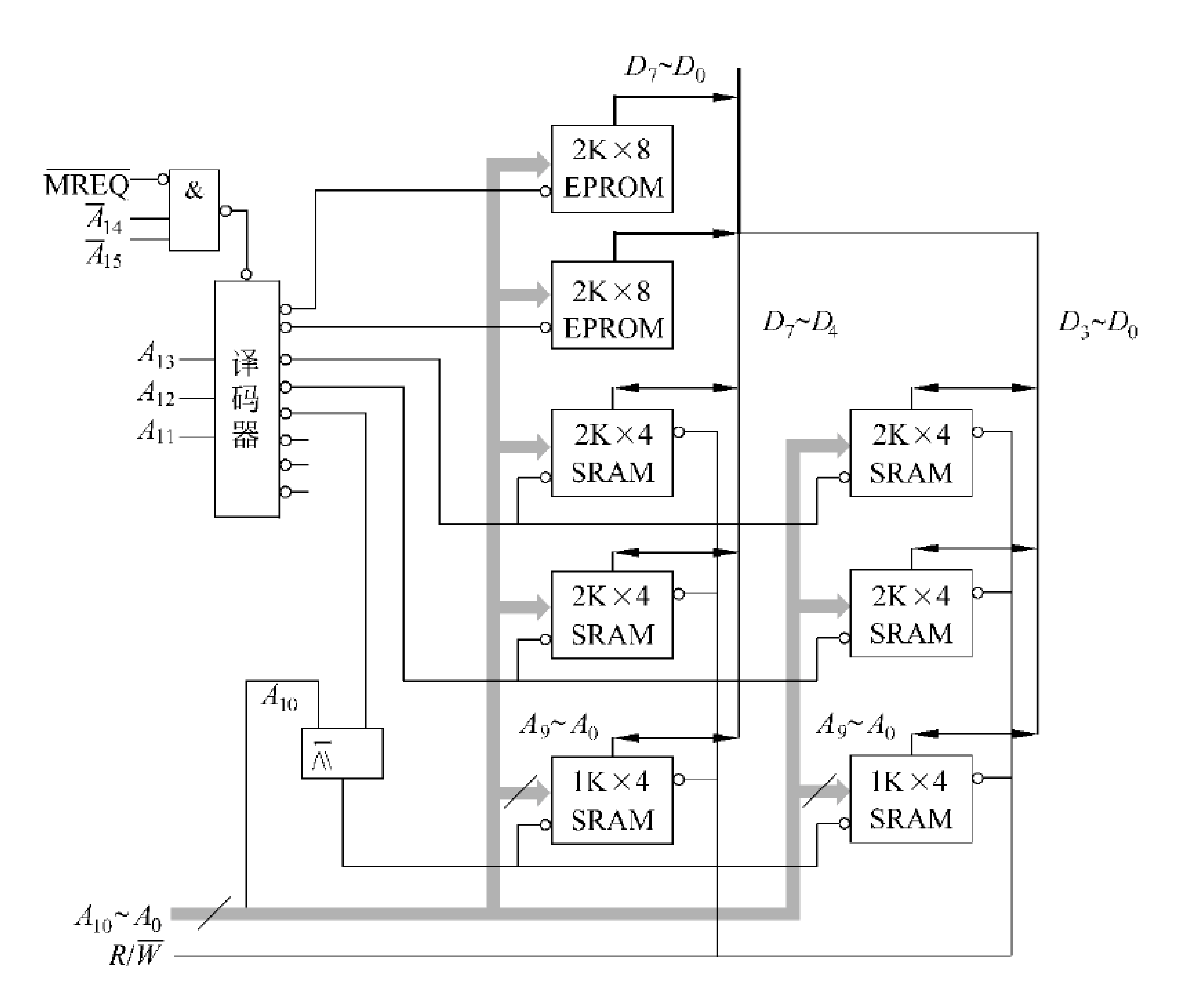


图 5-32　设计题 7 的存储器逻辑图

$SRAM_1+SRAM_2$　1000H～17FFH

$SRAM_3+SRAM_4$　1800H～1FFFH

$SRAM_5+SRAM_6$　2000H～23FFH

各片选逻辑式为：

$$CS_0=\overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}\ \overline{A_{11}}$$

$$CS_1=\overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}A_{11}$$

$$CS_2=\overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}A_{12}\ \overline{A_{11}}$$

$$CS_3=\overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}A_{12}A_{11}$$

$$CS_4=\overline{A_{15}}\ \overline{A_{14}}A_{13}\ \overline{A_{12}}\ \overline{A_{11}}\ \overline{A_{10}}$$

8. 根据图 5-26，可知 RAM(6264)芯片的容量为 8KB(13 根地址线，8 根数据线)，ROM(2732)芯片的容量为 4KB(12 根地址线，8 根数据线)。所以主存储器需要 2732 两片，6264 五片。各芯片的地址分配如图 5-33 所示。

该存储器的逻辑框图见图 5-34。

9. (1) 根据 Cache 容量 4096 字，得 Cache 地址为 12 位。根据块长为 4，且访存地址为字地址，得字块内地址为 2 位，且 Cache 共有 1024 块(4096÷4)。根据主存容量为 512K，得主存地址为 19 位。在直接映射方式下主存字块标记为 7 位(19－12)，主存地址格式如图 5-35(a)所示。

(2) 在全相联映射方式下，主存字块标记为 17 位(19－2)，主存地址格式如图 5-35(b)所示。

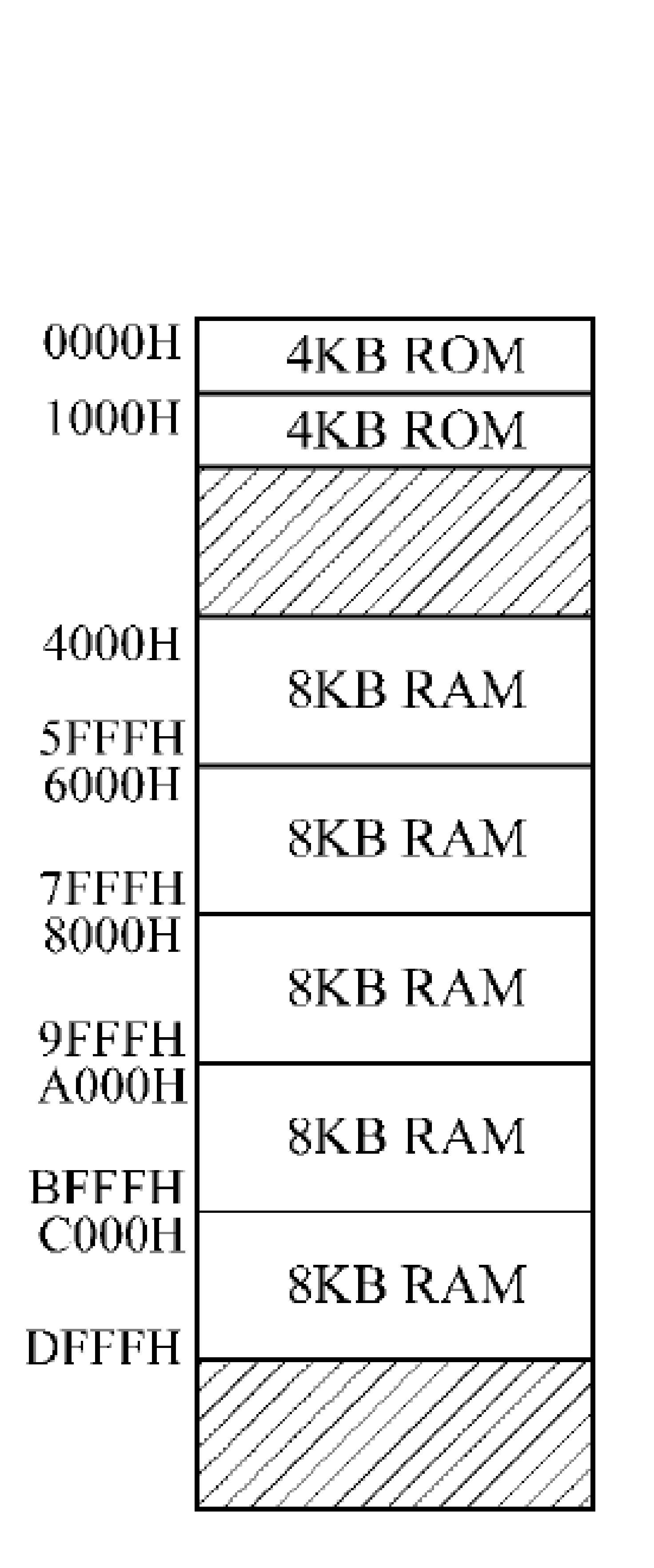


图 5-33　各芯片的地址分配

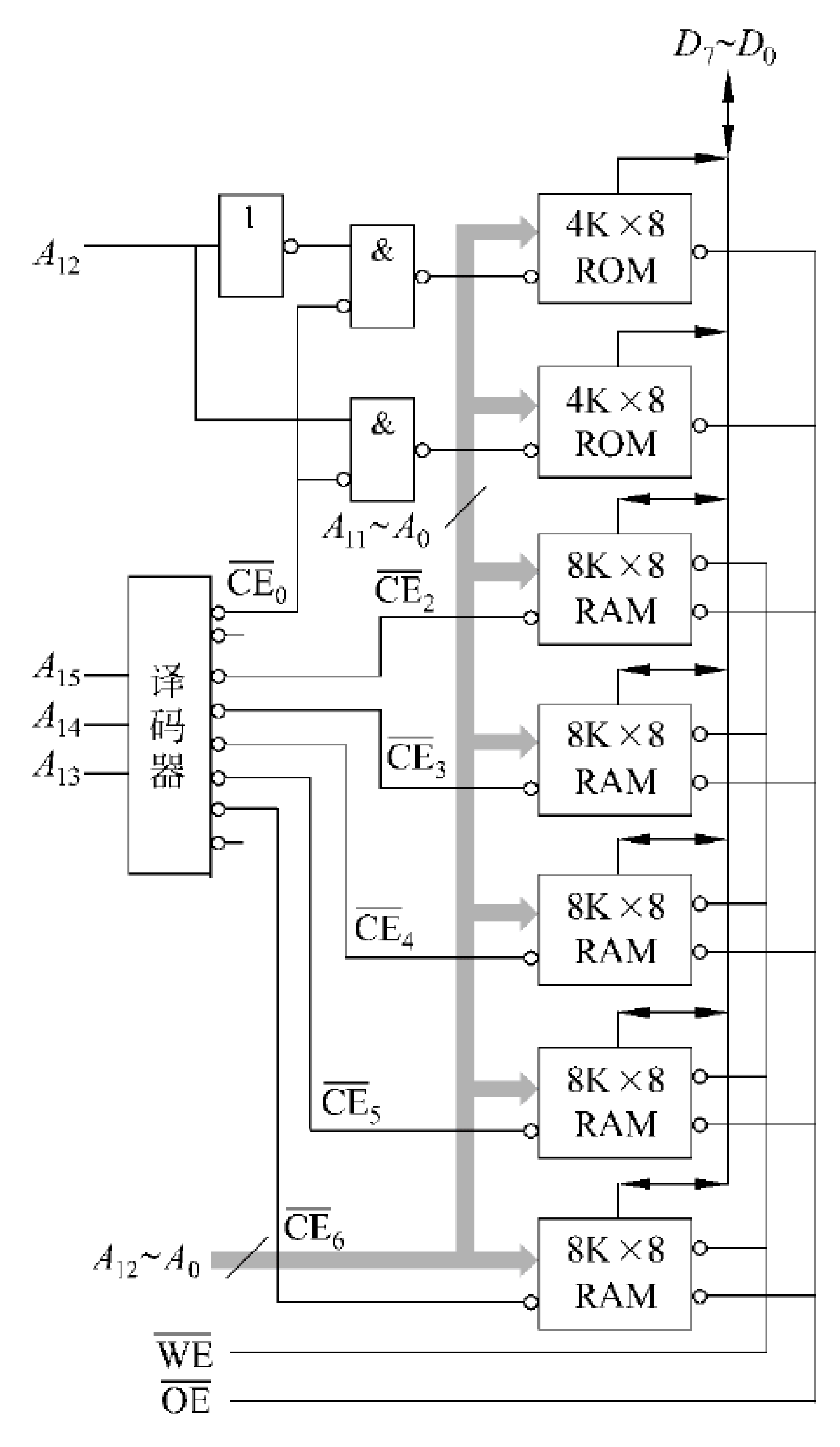


图 5-34　设计题 8 存储器的逻辑框图

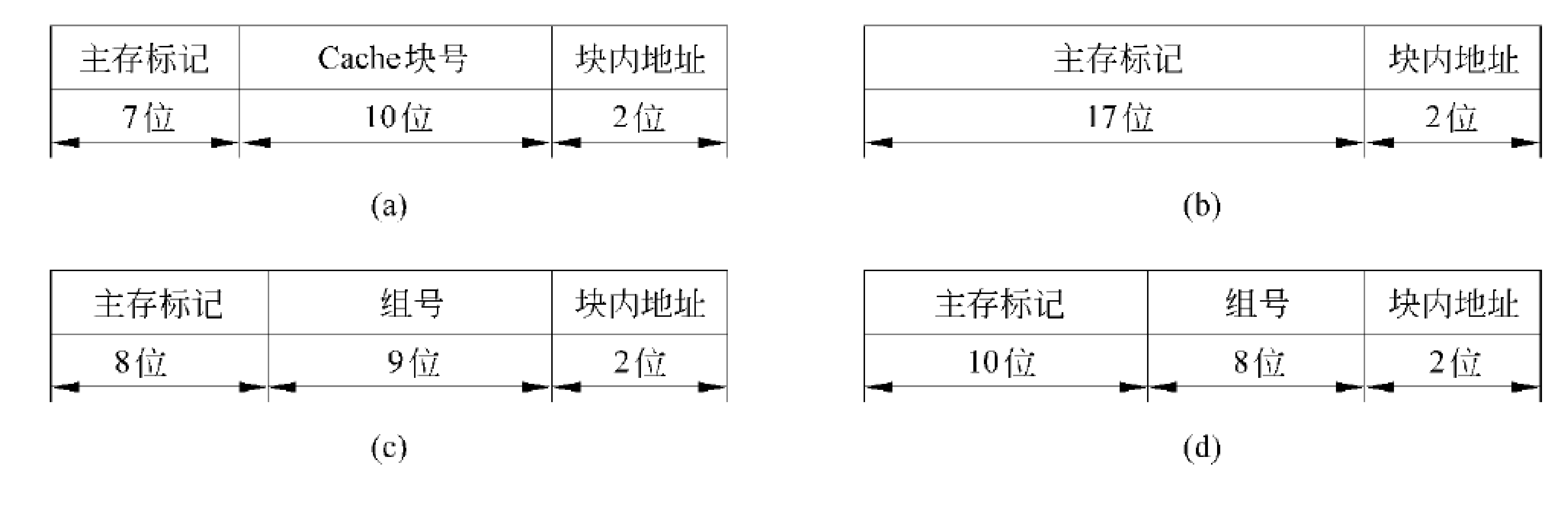


图 5-35　设计题 9 的主存地址格式

(3) 根据 2 路组相联的条件，一组内有 2 块，得 Cache 共分 512 组，主存字块标记为 8 位，其地址格式如图 5-35(c)所示。

(4) 若主存容量改为 512K×32 位，即双字宽存储器，块长不变，访存仍为字地址，则主存容量可写为 1024K×16 位，得主存地址 20 位。由 4 路组相联，得 Cache 共分为 256 组。主存字块标记为 10 位，地址格式如图 5-35(d)所示。

第 6 章 中央处理器

6.1 基本内容摘要

- 中央处理器的功能和组成
  - CPU 的功能

    指令流和数据流；

    数据流由指令流驱动。
  - CPU 中的主要寄存器

    通用寄存器；

    专用寄存器。
  - CPU 的组成
  - CPU 的主要技术参数

    字长；

    内部工作频率；

    外部工作频率。
- 控制器的组成和实现方法
  - 控制器的基本组成

    指令部件；

    时序部件；

    微操作信号发生器；

    中断控制逻辑。
  - 控制器的硬件实现方法

    组合逻辑型；

    存储逻辑型；

    组合逻辑和存储逻辑结合型。
- 时序系统与控制方式
  - 时序系统

    周期、节拍、脉冲。
  - 控制方式

同步控制方式；
异步控制方式；
联合控制方式。

◆ 指令运行的基本过程
取指令阶段；
分析取数阶段；
执行阶段。

◆ 指令的微操作序列

● 微程序控制原理

◆ 微程序控制的基本概念

◆ 微指令编码法
直接控制法；
最短编码法；
字段编码法。

◆ 微程序控制器的组成和工作过程

◆ 微程序入口地址的形成
由机器指令的操作码字段指出微程序的入口地址。

◆ 后继微地址的形成
增量方式；
断定方式。

● 控制单元的设计

◆ 简单的 CPU 模型

◆ 组合逻辑控制单元设计

◆ 微程序控制单元设计

● 流水线技术

◆ 重叠控制

◆ 先行控制原理

◆ 流水工作原理

● 精简指令系统计算机

◆ RISC 的特点和优势

◆ RISC 基本技术

● 微处理器中的新技术

◆ 超标量和超流水线技术

◆ EPIC 的指令级并行处理

◆ 超线程技术

◆ 双核与多核技术

6.2 重点难点梳理

1. 指令流和数据流

指令流和数据流都是程序运行的动态概念，它不同于程序中静态的指令序列，也不同于存储器中数据的静态分配序列。指令流指的是处理器执行的指令序列，数据流指的是根据指令操作要求依次存取数据的序列。

对指令流的控制主要包括指令流出的控制、指令分析与执行的控制、指令流向的控制。

对数据流的控制主要应包括对数据的流入与流出的控制；对数据变换、加工等操作的控制。

对于冯·诺依曼结构的计算机而言，它的数据流是根据指令流的操作而形成的，也就是说，数据流是由指令流来驱动的。

2. CPU 中专用寄存器

程序计数器(PC)又称为指令计数器或指令指针(IP)，用来存放指令地址或接着要执行的下一条指令地址。在程序开始执行前，将程序中第一条指令所在的主存单元地址送入PC，以便从第一条指令开始执行。在执行程序过程中，CPU 将自动修改 PC 的内容，使其保存将要执行的下一条指令的地址。由于大多数指令都是按顺序执行的，所以修改的过程通常只是简单地对 PC 增量计数，增量值取决于主存的编址方式，若主存按字节编址，则增量值为指令的字节数。如果是转移指令，则需要将形成的转移地址送至 PC 作为下一条指令的地址。因此，PC 应具有计数功能和接收代码的功能。可以让 PC 本身具有计数功能，也可以通过 ALU 实现加 1 计数。在后一种情况下，PC 实际上是单纯的寄存器。通过 PC 内容的不断更新，控制机器执行指令序列的进程。

指令寄存器(IR)用来存放现行指令。当执行一条指令时，首先从主存将指令取出送到指令寄存器中，直到这一条指令执行结束再放入下一条指令。为了提高指令的执行速度，现在大多数计算机都将指令寄存器扩充为指令队列(指令栈)，允许预取若干条指令。

存储器地址寄存器(MAR)和存储器数据寄存器(MDR)是 CPU 和主存之间进行数据交换的一对接口。MAR 用来接收指令地址(PC 的内容)、操作数地址或结果数据地址，以确定要访问的单元。MDR 有时也称为数据缓冲寄存器(MBR)，它是向主存写入数据或从主存读出指令或数据的缓冲部件。

状态标志寄存器(PSWR)用来存放程序状态字，程序状态字的各位表征程序和机器运行的状态，它是参与控制程序执行的重要依据之一，主要包括两部分内容：一是状态标志，例如进位标志(C)、结果为零标志(Z)等，许多指令的执行将会自动修改这些标志位；二是控制标志，例如中断标志、陷阱标志等。

3. 控制器的基本组成和硬件实现方法

控制器主要由以下几部分组成：

(1) 指令部件。包括程序计数器(PC)、指令寄存器(IR)、指令译码器(ID)、地址形成部件等。

(2) 时序部件。包括脉冲源、启停控制逻辑、节拍信号发生器等。

(3) 微操作信号发生器。一条指令的取出和执行可以分解成很多最基本的操作，这种

最基本的不可再分割的操作称为微操作。微操作信号发生器也称为控制单元(CU)。真正控制各部件工作的微操作信号是由指令部件提供的操作信号、时序部件提供的时序信号、被控制功能部件所反馈的状态及条件信号综合形成的。不同的机器指令具有不同的微操作序列。

(4) 中断控制逻辑。控制中断处理的硬件逻辑。

控制器的输入包括时序信号、机器指令操作码、各部件状态反馈信号等,输出的微操作控制信号又可细分为 CPU 内的控制信号和送至主存或外设的控制信号。根据产生微操作控制信号的方式不同,控制器可分为组合逻辑型、存储逻辑型、组合逻辑与存储逻辑结合型3种,它们的根本区别在于微操作信号发生器的实现方法不同,而控制器中的其他部分基本上大同小异。

4. 组合逻辑(硬布线)控制和微程序控制的比较

组合逻辑控制和微程序控制的主要区别在于微操作信号发生器(控制单元)的实现方法不同。具体说明如下:

(1) 组合逻辑控制的控制功能是由组合逻辑电路控制实现的,由于各个微操作控制信号的逻辑表达式的繁简程度不同,由此组成的控制电路零乱、复杂。而微程序控制的控制功能是由存放微程序的控制存储器和存放当前正在执行的微指令的寄存器直接控制实现的,控制电路比较规整。

(2) 对组合逻辑控制来说,因为所有控制信号的逻辑表达式用硬连线固定下来,当需要修改和增加指令时就很麻烦,有时甚至可能需要重新进行设计。而在微程序控制中,各条指令的微操作控制信号的差别仅反映在控制存储器的内容上,如果想扩展和改变机器的功能,只需在控制存储器中增加新的微指令或修改某些原来的微指令即可。

(3) 在同样的半导体工艺条件下,组合逻辑控制的速度比微程序控制的速度快。这是因为组合逻辑控制的速度主要取决于逻辑电路的延迟,而微程序控制执行每条微指令都要从控存中读取,影响了速度。

5. 指令周期和机器周期

指令周期是指一条指令从取出到执行完成所需要的全部时间。由于各种指令的操作类型不同、寻址方式不同,因此,它们的指令周期也不相同。例如,访存指令与不访存指令的指令周期不同;一条加法指令与一条乘法指令的指令周期也不同。

机器周期又称为 CPU 周期。通常把一个指令周期划分为若干个机器周期,每个机器周期完成一个基本操作。一般机器的 CPU 周期有取指周期、取数周期、执行周期、中断周期等。所以有:

$$\text{指令周期} = i \times \text{机器周期}$$

不同的指令周期中所包含的机器周期数差别可能很大。一般情况下,一条指令所需的最短时间为两个机器周期:取指周期和执行周期。

为了使 CPU 能明确当前处于何种机器周期,每个机器周期都应有一个与之对应的周期状态触发器。机器运行在不同的机器周期时,其对应的周期状态触发器被置“1”。显然,在机器运行的任何时刻只能处于一种周期状态,因此,有一个且仅有一个触发器被置“1”。当某个触发器为“1”时,表示机器进入处理指令的对应阶段,并执行该阶段的微操作序列。

由于 CPU 内部的操作速度较快,而 CPU 访问主存所花费的时间较长,所以许多计算

机系统往往以主存的工作周期(存取周期)为基础来规定CPU周期,以便两者的工作能配合协调。CPU访问主存也就是一次总线传送,故在微型计算机中称为总线周期。

6. 多级时序系统

多级时序系统将时序关系划分为几个层次,常见的有机器周期、节拍、脉冲3级时序系统。在时序系统中一般都不为指令周期设置完整的时间标志信号,因此,一般不将指令周期视为时序的一级。

一个机器周期的工作可能需要分成几步按照一定顺序完成,为此,将一个机器周期又分为若干个相等的时间段,每一个时间段对应一个节拍。在一个节拍内常常设置一个或几个工作脉冲,以实现对某些微操作定时。具体机器设置的脉冲数根据需要而有所不同,有的机器只在节拍的末尾设置一个定时脉冲,其前沿用于结果寄存器的接数,后沿则实现周期切换;也有的机器在一个节拍中先后设置几个定时脉冲,分别用于清除、接数和周期切换等目的。

7. 指令运行的基本过程

一条指令运行过程可以分为3个阶段:取指令阶段、分析取数阶段和执行指令阶段。

1) 取指令

取指令阶段完成的任务是将现行指令从主存中取出来并送至指令寄存器中去。具体操作如下:

(1) 将程序计数器(PC)中的内容送至存储器地址寄存器(MAR),并送至地址总线(AB)。

(2) 向存储器发读命令。

(3) 从主存中取出的指令通过数据总线(DB)送到存储器数据寄存器(MDR)。

(4) 将MDR的内容送至指令寄存器(IR)中。

(5) 将PC的内容递增,为取下一条指令做好准备。

以上这些操作对任何一条指令来说都是必须要执行的操作,所以称为公共操作。完成取指令阶段任务的时间称为取指周期。

2) 分析取数阶段

取出指令后,机器立即进入分析取数阶段,指令译码器可识别和区分不同的指令类型。由于各条指令功能不同,寻址方式也不同,所以分析取数阶段的操作各不相同。

对于无操作数指令,只要识别出是哪一条具体的指令即可转执行阶段。而对于带操作数指令就需要读取操作数,首先要计算出操作数的有效地址,如果操作数在通用寄存器中,则不需要再访问主存;如果操作数在主存中,则要到主存中去取数。对于不同的寻址方式,有效地址的计算方法是不同的,有时要多次访问主存才能取出操作数来。另外,单操作数指令和双操作数指令由于需要的操作数的个数不同,分析取数阶段的操作也不同。

3) 执行阶段

执行阶段完成指令规定的各种操作,形成稳定的运算结果,并将其存储起来。

计算机的基本工作过程可以概括成为取指令、取数、执行指令,然后再取下一条指令……直至遇到停机指令或外来的干预为止。

8. 指令的微操作序列

控制器在实现一条指令的功能时,总要把每条指令分解成为一系列时间上先后有序的

最基本、最简单的微操作,即微操作序列。微操作序列是与 CPU 的内部数据通路密切相关的,不同的数据通路就有不同的微操作序列。事实上,要写出指令的微操作序列,首先需要给出相应的 CPU 结构和数据通路图,严格按要求建立起信息在计算机各部件之间流动的时间和空间关系,而不是凭空瞎编。如果 CPU 内部采用单总线结构,还要考虑总线冲突的问题,相应的微操作控制信号必须与给出的数据通路结构一致,且时序上要有先后顺序。

例如,某假想机的结构如图 6-1 所示,其中 R_0～R_3 为通用寄存器,A、B 为暂存器。

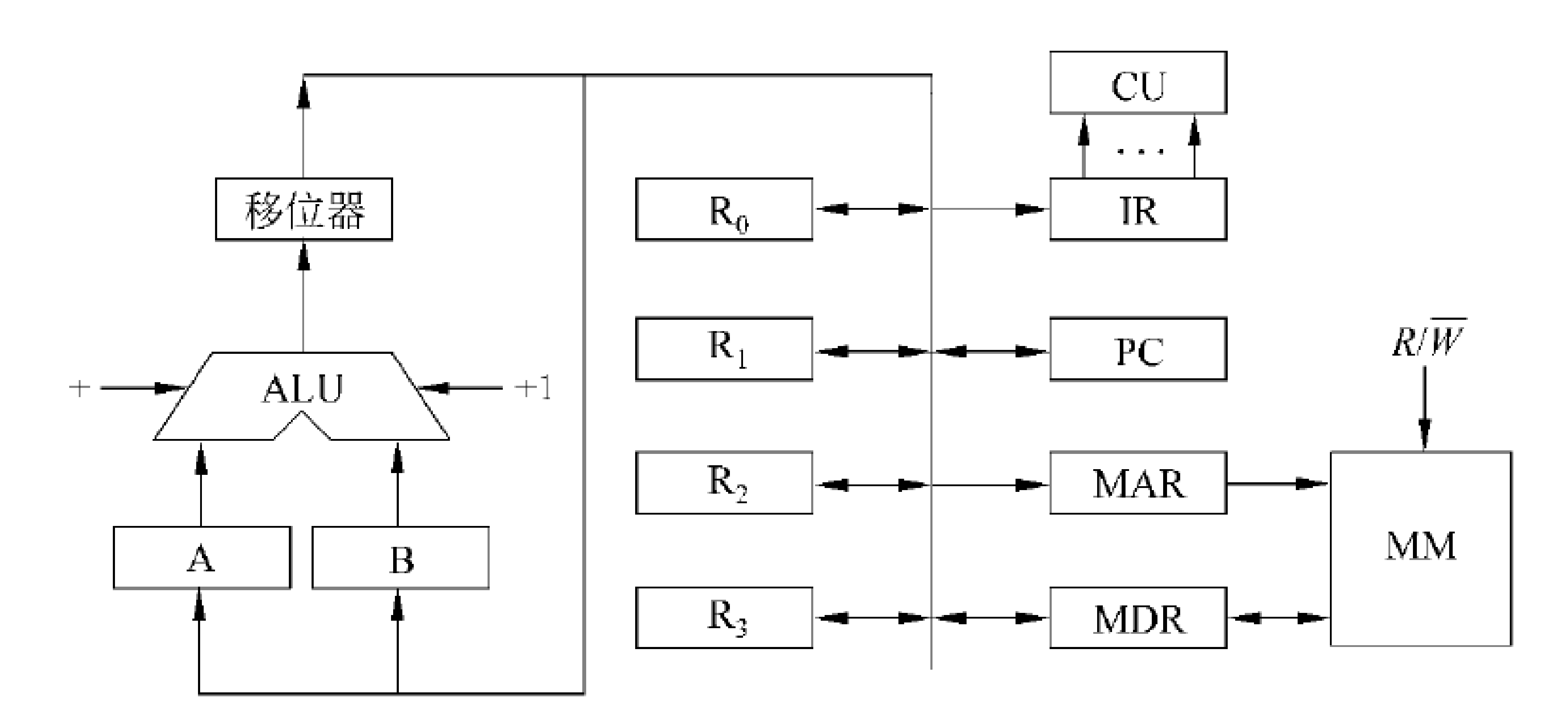


图 6-1　某假想机的结构

若有 3 条传送(MOV)指令:

```
MOV R0,R1          ;(R0)→R1
MOV (R0), R1       ;((R0))→R1
MOV R0, (R1)       ;(R0)→ (R1)
```

这 3 条指令的操作流程如图 6-2 所示。

加法指令 ADD R_2,(R_1)完成((R_1))+(R_2)→(R_1)操作,其指令的操作流程如图 6-3 所示。

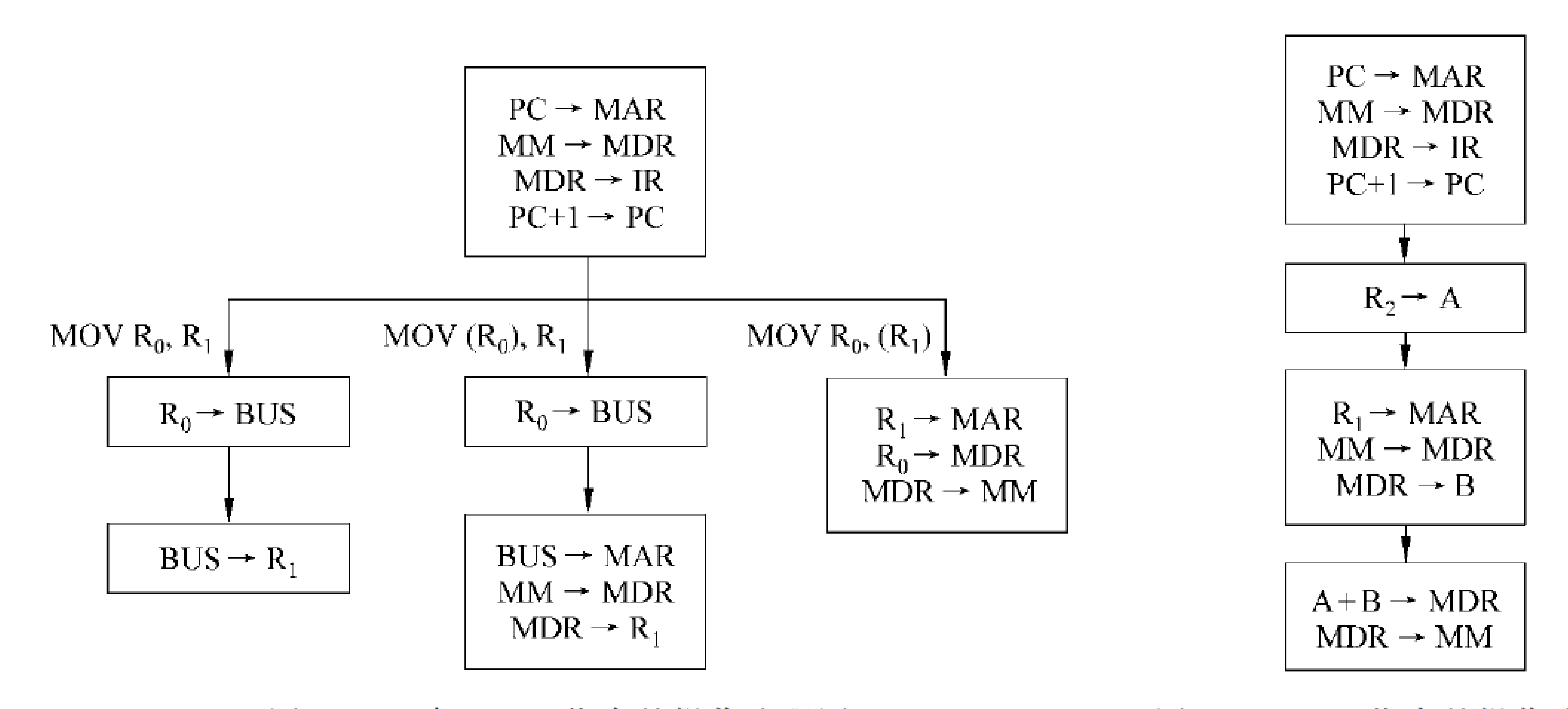


图 6-2　3 条 MOV 指令的操作流程图　　　　图 6-3　ADD 指令的操作流程图

从图 6-2 和图 6-3 中可以看出,每条指令在取指令阶段(取指周期)完成的操作是相同的,其任务是将现行指令从主存中取出来并送至指令寄存器中。在指令被取出来之前并不

清楚这是一条什么样的指令,所以取指令阶段所做操作与具体指令无关。

9. 微命令和微操作

在微程序控制的计算机中,将控制部件向执行部件发出的各种控制命令称为微命令,它是构成控制序列的最小单位。例如,打开或关闭某个控制门的电位信号,某个寄存器的打入脉冲,等等。因此,微命令是控制计算机执行部件(如运算器、存储器、外部设备)完成某个微操作的命令。微操作是计算机执行部件接收微命令后所执行的最基本的操作。一条机器指令可以分解成一个微操作序列,这些微操作是计算机中最基本的、不可再分解的操作。

微命令和微操作是一一对应的。微命令是微操作的控制信号,微操作是微命令控制的操作过程,在计算机内部实质上是同一个信号。

微命令有兼容性和互斥性之分,兼容性微命令是指那些可以同时产生,共同完成某一些微操作的微命令;而互斥性微命令是指在机器中不允许同时出现的微命令。兼容和互斥都是相对的,一个微命令可以和一些微命令兼容,和另一些微命令互斥。对于单独一个微命令,谈论其兼容和互斥都是没有意义的。

10. 微程序控制的计算机涉及的两个层次

微程序控制的计算机涉及两个层次:一个是机器语言或汇编语言程序员所看到的传统机器层,包括机器指令、工作程序、主存储器;另一个是机器设计者看到的微程序层,包括微指令、微程序和控制存储器。

机器指令是用户编程的基本单位,它表明 CPU 能够完成的一项基本功能。微指令则是为实现机器指令中某一步操作的若干个微命令的集合。一条机器指令对应由若干条微指令系列组成的微程序,机器指令由微指令进行解释并执行。

程序是由机器指令构成的,对用户程序而言,为某项任务而编制并存放在主存中,允许修改。微程序由微指令构成,一条机器指令对应一个微程序,所以微程序是用于描述机器指令的。微程序是在设计计算机时预先编制好的,并存入控制存储器中,通常不允许用户修改。但也有某些机器向用户提供了微程序设计能力,允许用户自己编制微程序,扩充所需要的功能。

主存储器中存放的是系统程序和用户程序,容量很大。控制存储器中存放的是对应于机器指令系统的全部微程序,控制实现机器的整个指令系统,容量有限。

11. 微指令操作控制字段的编码法

1) 直接控制法

在微指令的操作控制字段中,每个独立的二进制位代表一个微命令,该位为“1”表示这个微命令有效,为“0”则表示这个微命令无效。微命令的产生不必经过译码,所需的控制信号直接送到相应的控制点。

这种方法结构简单,并行性强,操作速度快,但是微指令字太长,使控存单元的位数过多,而且在给定的任何一个微指令中,往往仅使用了部分微命令,造成信息利用率下降。

2) 最短编码法

最短编码法将所有的微命令统一编码,每条微指令只定义一个微命令。若微命令的总数为 N,操作控制字段的长度为 L,则最短编码法应满足下列关系式:

$$L \geqslant \log_2 N$$

最短编码法的微指令字长最短,但要通过一个微命令译码器译码以后才能得到需要的

微命令。这种方法在同一时刻只能产生一个微命令,不能充分利用机器硬件所具有的并行性,使得机器指令对应的微程序变得很长,而且对于某些要求在同一时刻同时动作的组合性微操作将无法实现。

3) 字段编码法

这是前述两种编码法的一个折中的方法,既具有两者的优点,又克服了它们的缺点。这种方法将操作控制字段分为若干个小段,每段内采用最短编码法,段与段之间采用直接控制法。这种方法又可进一步分为字段直接编码法和字段间接编码法。

12. 字段编码法的分段原则

在字段编码法中,操作控制字段的分段并非是任意的,必须要遵循如下的原则:

(1) 应把互斥性的微命令分在同一段内,兼容性的微命令分在不同段内。这样不仅有助于提高信息的利用率,缩短微指令字长,而且有助于充分利用硬件所具有的并行性,加快执行的速度。

(2) 应与数据通路结构相适应。

(3) 每个小段中包含的信息位不能太多,否则将增加译码线路的复杂性和译码时间。

(4) 一般每个小段还要留出一个状态,表示本字段不发出任何微命令。因此当某字段的长度为3位时,最多只能表示7个互斥的微命令,通常用000表示不操作。

13. 微程序控制计算机的基本结构

微程序控制计算机与组合逻辑控制计算机的主要差别在于控制单元的不同,微程序控制计算机的控制单元部分用一个完整的"存储系统"(包括控制存储器、微指令寄存器、微地址寄存器等)代替了组合逻辑控制网络。

注意:这里所说的"存储系统"与第5章中提到的存储系统不是一回事,这个"存储系统"属于中央处理器的一部分。

控制存储器(CM)是微程序控制器的核心部件,用来存放微程序。其性能(包括容量、速度、可靠性等)与计算机的性能密切相关。

微指令寄存器(μIR)用来存放从CM取出的正在执行的微指令,它的位数同微指令字长相等。

微地址形成部件用来产生初始微地址和后继微地址,以保证微指令的连续执行。

微地址寄存器(μMAR)用于接收微地址形成部件送来的微地址,为在CM中读取微指令作准备。

14. 微程序控制器的工作过程

在微程序控制计算机中,机器指令是通过读取微程序并由微程序所包含的微命令来解释执行的。每条机器指令所对应的微程序已经编制好,并放入CM中,如图6-4所示。

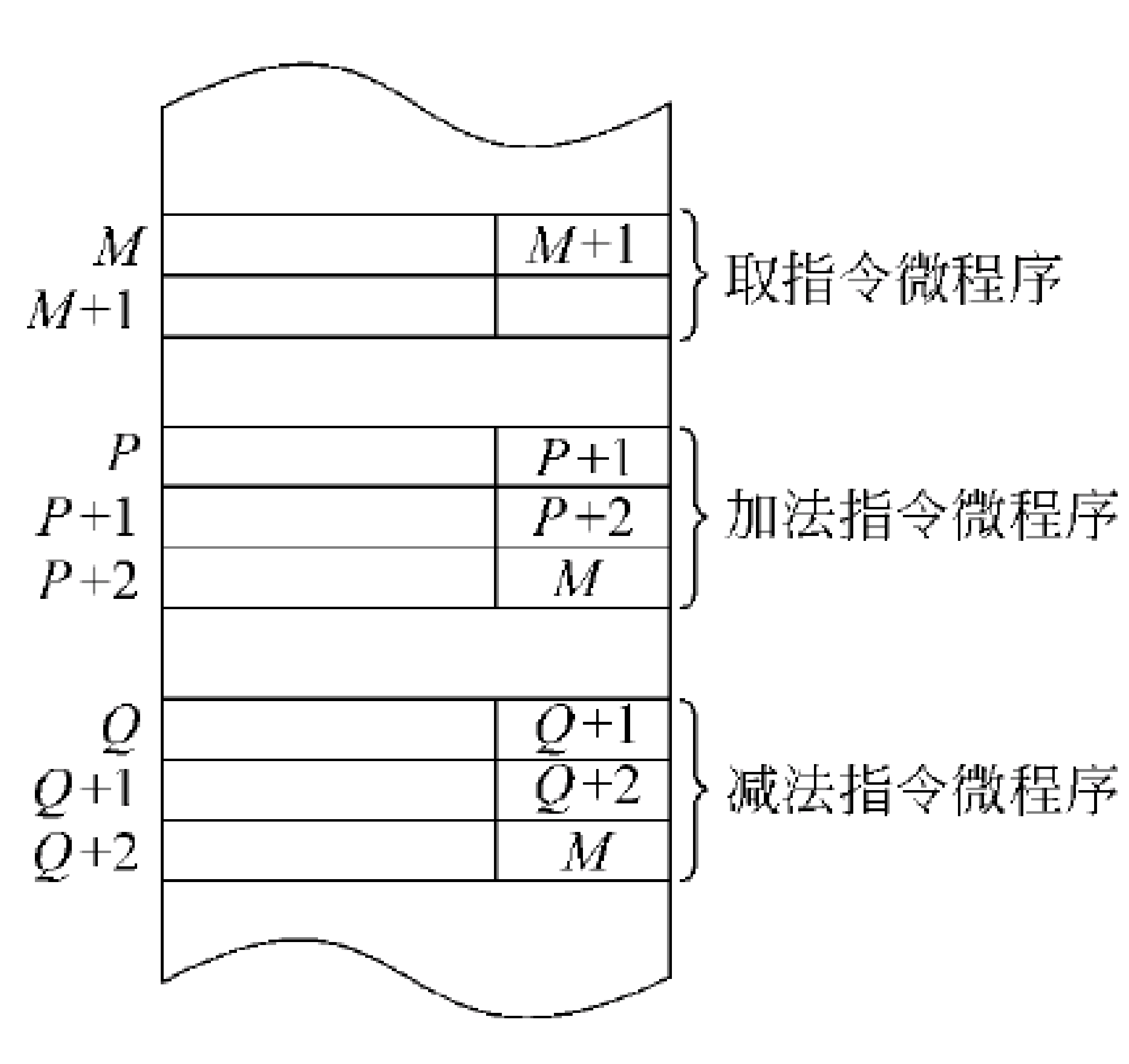


图6-4 不同机器指令对应的微程序

(1) 执行取指令微程序。在机器开始运行时,自动地将取指令微程序的入口微地址 M 送 μMAR,并从CM中读出相应的微指令送入μIR。取指令微程序中各个微命令使CPU访问主存,读取机器指令送入指令寄存器(IR),并修改程序计数器(PC)的内

容。当取指令微程序执行完后，从主存中取出的机器指令就已存入IR中了。

(2) 根据机器指令的操作码字段通过微地址形成部件产生该机器指令所对应的微程序的入口地址(如图6-4中的 P 和 Q)，并送入 μMAR。

(3) 从CM中逐条取出对应微程序的各条微指令并执行，直至该机器指令所对应的微程序执行完为止。

(4) 执行完对应的一个微程序后，微地址又回到取指令微程序的入口地址 M，继续第(1)步。

由于任何一条机器指令的取指令操作都是相同的，因此可以将取指令操作抽出来编成一个独立的微程序，同理，也可以编出对应间址周期的微程序和中断周期的微程序。这样，控制存储器中的微程序个数应等于指令系统中的机器指令数再加上对应取指、间址和中断周期等公用的微程序数。若指令系统中具有 n 种机器指令，则控制存储器中的微程序数至少有 $n+1$ 个。

15. 微程序入口地址的形成

由于每条机器指令都需要取指操作，所以将取指操作编制成一段公用微程序，通常安排在控存的0号或1号单元开始的一段CM空间中。

每条机器指令对应一个微程序，当执行公用的取指令微程序从主存中取出一条机器指令送到IR后，由机器指令的操作码字段指出微程序的入口地址(初始微地址)，这是一种多分支(或多路转移)的情况。

根据机器指令的操作码形成微程序入口地址的最简单方式是一级功能转换。如果机器指令操作码字段的位数和位置固定，可以根据指令操作码一次转移到相应的微程序入口，采取的方法是直接使操作码与微地址码的部分位相对应。例如，操作码用 θ 表示，微程序的入口地址可以为 $\theta\times\cdots\times$。若MOV指令的操作码为0000，则MOV指令的微程序入口地址为 $0000\times\cdots\times$；ADD指令的操作码为0001，则ADD指令的微程序入口地址为 $0001\times\cdots\times$……由此可见，相邻两个微程序的入口地址相差 n 个单元，即每个微程序最多可以由 n 条微指令组成，如果不足 n 条就让有关单元空闲着。n 与微地址码 $\times\cdots\times$ 的位数有关，$n=2^{\text{位数}}$。

16. 后继微地址的形成

找到初始微地址之后，可以开始执行微程序，每条微指令执行完毕都要根据要求形成后继微地址。后继微地址的形成方法对微程序编制的灵活性影响很大，它主要有两大基本类型：增量方式和断定方式。

1) 增量方式(顺序-转移型微地址)

这种方式和机器指令的控制方式类似，也有顺序执行、转移和转子之分。顺序执行时后继微地址就是现行微地址加上一个增量(通常为“1”)；转移或转子时，由微指令的顺序控制字段产生转移微地址。因此，在微程序控制器中应当有一个微程序计数器(μPC)。为了降低成本，一般情况下都是将微地址寄存器μMAR改为具有计数功能的寄存器，以代替μPC。增量方式的优点是简单、易于掌握、编制微程序容易，每条机器指令所对应的一段微程序一般安排在CM的连续单元中；其缺点是这种方式不能实现多路转移，因而不利于提高微程序的执行速度。

2) 断定方式

断定方式的后继微地址可由微程序设计者指定，或者根据微指令所规定的测试结果直

接决定后继微地址的全部或部分值。

这是一种直接给定与测试断定相结合的方式，微指令的顺序控制字段一般由两部分组成：非测试段(下址字段)和测试段，采用断定方式的微指令格式如图 6-5 所示。

非测试段由设计者指定，一般对应微地址的高位部分，用来指定后继微地址在 CM 中的某个区域内；测试段将通过微地址形成部件修改微地址寄存器的适当位数(一般对应微地址的低位部分)，从而实现微程序的分支转移，这相当于在指定区域内断定具体的分支。所依据的测试状态可能是指定的开关状态、指令操作码、状态字等。测试段如果只有 1 位，则微地址将产生 2 个分支；若有 2 位，则最多可产生 4 个分支；以此类推；测试段为 n 位最多可产生 2^n 个分支。

下面以一个简单的例子来说明断定方式下 2～4 路转移的情况，转移原理如图 6-6 所示。μMAR 的最低两位 C、D 分别由以下两组状态来决定，一组状态{0、1、T_1、T_2}来决定 C 位，另一组状态{0、1、K_1、K_2}来决定 D 位。T_1、T_2、K_1、K_2 表示运算结果、测试结果或某些硬件状态。

操作控制字段	顺序控制字段	
微操作码	测试判别	下址

图 6-5　断定方式的微指令格式

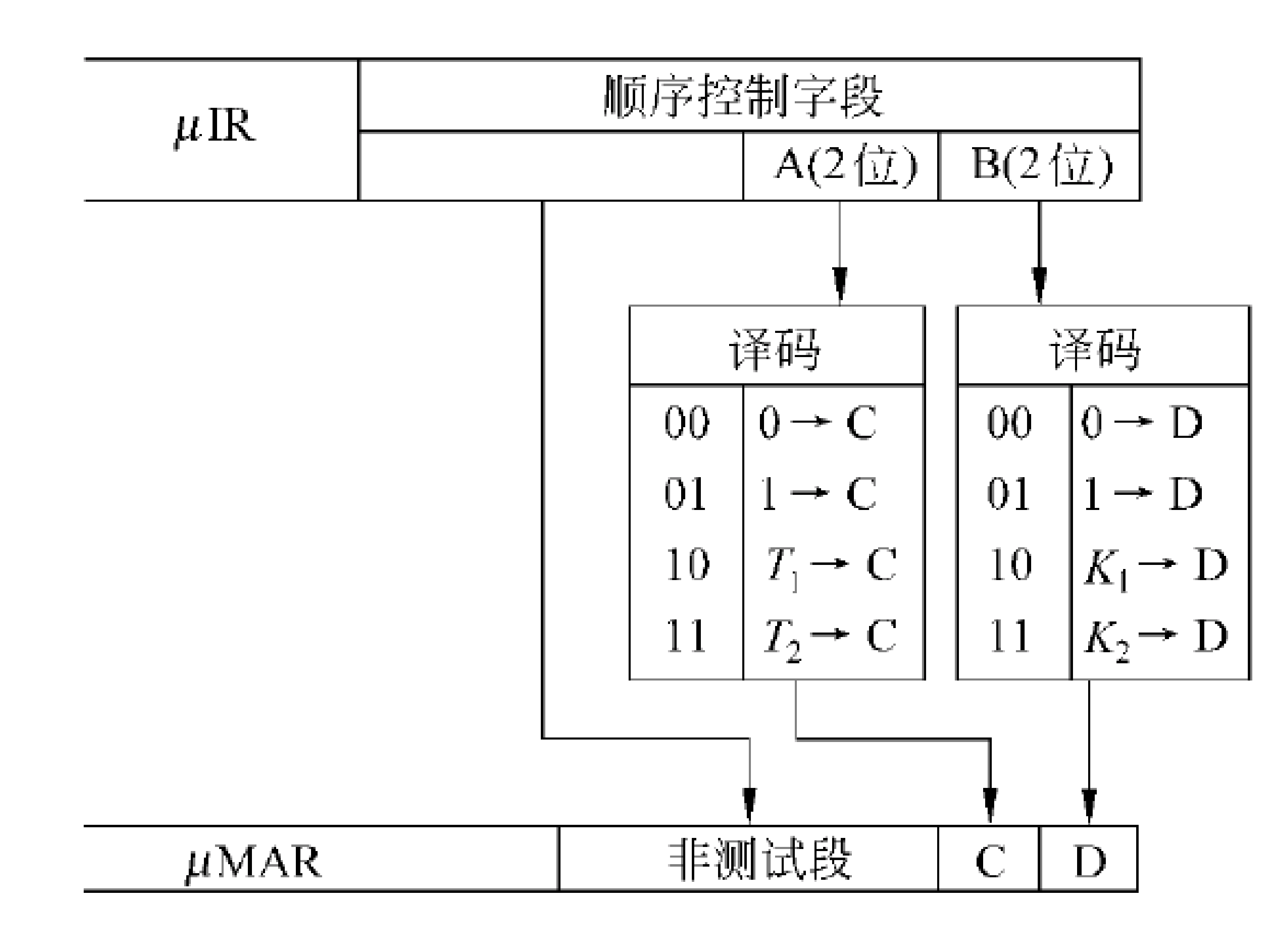


图 6-6　断定方式转移地址的产生

2 路转移时，有以下两种情况：

(1) A 字段为 00 或 01，B 字段为 10 或 11，此时后继微地址根据 K_i 的状态决定。当非测试段为 0101 时，A 选 00 时，后继微地址为 $01010K_i$，如 $K_i=0$，则后继微地址为 010100，如 $K_i=1$，则后继微地址为 010101。

(2) A 字段为 10 或 11，B 字段为 00 或 01，此时后继微地址根据 T_j 的状态决定。

4 路转移，T_i 和 K_j 的 4 种可能的组合状态决定了 4 个转移地址，即 μMAR 的最低两位为 00、01、10、11。

3 路转移实际上是令 4 路转移的 4 个可能的后继微地址中有 2 个是相同的。

若要实现 4 路以上的转移，则应在微指令的顺序控制字段中增加位数，以改变 μMAR 中更多位的值。如要实现 64 路转移，则要改变 μMAR 中低 6 位的微地址码。

17. 水平型微指令和垂直型微指令

微指令有垂直型和水平型之分。

水平型微指令是指一次能定义并且能并行执行多个微命令的微指令。采用水平型微指令编制微程序时，由于微指令的并行操作能力强、效率高、编制的微程序比较短，因此，微程

序的执行速度比较快，控制存储器的纵向容量小、灵活性强。其缺点是微指令字比较长，明显地增加了控制存储器的横向容量。同时，水平型微指令与机器指令差别很大，只有熟悉机器的数据通路结构、微命令系统以及指令执行过程的人，才有可能编制出理想的微程序，一般用户不易掌握。因为水平型微程序设计是面对微处理器内部逻辑控制的描述，所以把这种微程序设计方法称为硬方法。

垂直型微指令是指一次只能执行一个微命令的微指令。采用垂直型微指令编制微程序时，只需注意微指令的功能，而对数据通路的细节不用过多地考虑，这是因为垂直型微指令与机器指令相似，容易掌握和利用，编程比较简单；同时，垂直型微指令字较短，使控制存储器的横向容量少。其缺点是，用垂直型微指令编制微程序要使用较多的微指令，微程序较长，因此，微程序执行速度慢，且要求控制存储器的纵向容量大。因为垂直型微程序设计是面向算法的描述，所以把这种微程序设计方法称为软方法。

18. CISC 和 RISC 的区别

基于复杂指令系统设计的计算机称为复杂指令系统计算机（CISC），基于精简指令系统设计的计算机称为精简指令系统计算机（RISC）。

RISC 的中心思想是要求指令系统简化，尽量使用寄存器-寄存器操作指令，除去访存指令（Load 和 Store）外，其他指令的操作均在单周期内完成，指令格式力求一致，寻址方式尽可能减少，并提高编译的效率，最终达到加快机器处理速度的目的。表 6-1 所示为 CISC 和 RISC 的区别。

表 6-1 CISC 和 RISC 的区别

指　　标	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于 200 条	一般小于 100 条
指令字长	不固定	等长
寻址方式	一般大于 4	一般小于 4
可访存指令	不加限制	只有 Load/Store 指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成
通用寄存器数量	较少	多
控制方式	绝大多数为微程序控制	绝大多数为硬布线控制

19. 流水线控制方式

流水线是将一个较复杂的处理过程分成 m 个复杂程度相当、处理时间大致相等的子过程，每个子过程由一个独立的功能部件来完成，处理对象在各子过程连成的线路上连续流动。在同一时间，m 个部件同时进行不同的操作，完成对不同子过程的处理，每隔 $\Delta t = T/m$ 就可以处理一个任务。

吞吐率 TP 指的是流水线机器在单位时间里能流出的任务数或结果数。

如果流水线各段的经过时间不同时，流水线的最大吞吐率

$$\mathrm{TP}_{\max} = \frac{1}{\max\{\Delta t_1, \cdots, \Delta t_i, \cdots, \Delta t_n\}}$$

它受限于流水线中最慢子过程经过的时间。流水线中经过时间最长的子过程称为瓶颈子过程。

流水线的实际吞吐率 TP 一般明显低于最大吞吐率 TP_{max}。设一个 m 段流水线的各段经过时间均为 Δt_0，则需要 $T_0=m\Delta t_0$ 的流水建立时间，之后每隔 Δt_0 就可流出一条指令，完成 n 个任务的解释共需时间 $T=m\Delta t_0+(n-1)\Delta t_0$，流水线的实际吞吐率为：

$$TP=\frac{n}{m\Delta t_0+(n-1)\Delta t_0}$$

6.3 典型例题详解

【例 6.1】 CPU 中专用寄存器有哪几个，各自的功能是什么？简述在一条指令的执行过程中这些寄存器各自将起什么作用(以一地址的算术运算指令为例)。

解：专用寄存器是专门用来完成某一种特殊功能的寄存器。CPU 中的专用寄存器有：

程序计数器(PC)，用来存放正在执行的指令地址或接着要执行的下一条指令地址。

指令寄存器(IR)，用来存放从存储器中取出的待执行的指令。

存储器地址寄存器(MAR)，用来接收来自程序计数器(PC)的指令地址或来自运算器的操作数地址。

存储器数据寄存器(MDR)，向主存写入数据或从主存读出指令或数据的缓冲部件。

状态标志寄存器(PSWR)，用来存放程序状态字。

若以一地址加 1 指令(INC)为例，可以看到各寄存器的作用如表 6-2 所示。

表 6-2　专用寄存器的作用

专用寄存器	取指周期	分析周期	执行周期
PC	(PC)→MAR	—	(PC)+1
IR	指令→MDR→IR	—	—
MAR	指令地址→MAR	A→MAR	—
MDR	指令→MDR	(A)→MDR	(A)+1→MDR
PSW	—	—	根据运算结果设置标志位

【例 6.2】 指令和数据都存放在主存，如何识别从主存储器中取出的是指令还是数据？

解：指令和数据都存放在主存，它们都以二进制代码形式出现，区分的方式有：

(1) 从主存中取出的机器周期不同。取指周期取出的是指令，分析取数或执行周期取出的是数据。

(2) 取指令和取数据时地址的来源不同。指令地址来源于程序计数器 PC，数据地址来源于地址形成部件。

【例 6.3】 CPU 结构如图 6-7 所示，其中包括一个累加寄存器 AC，一个状态寄存器和其他 4 个寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向。

(1) 标明 4 个寄存器的名称。

(2) 简述取指令的数据通路。

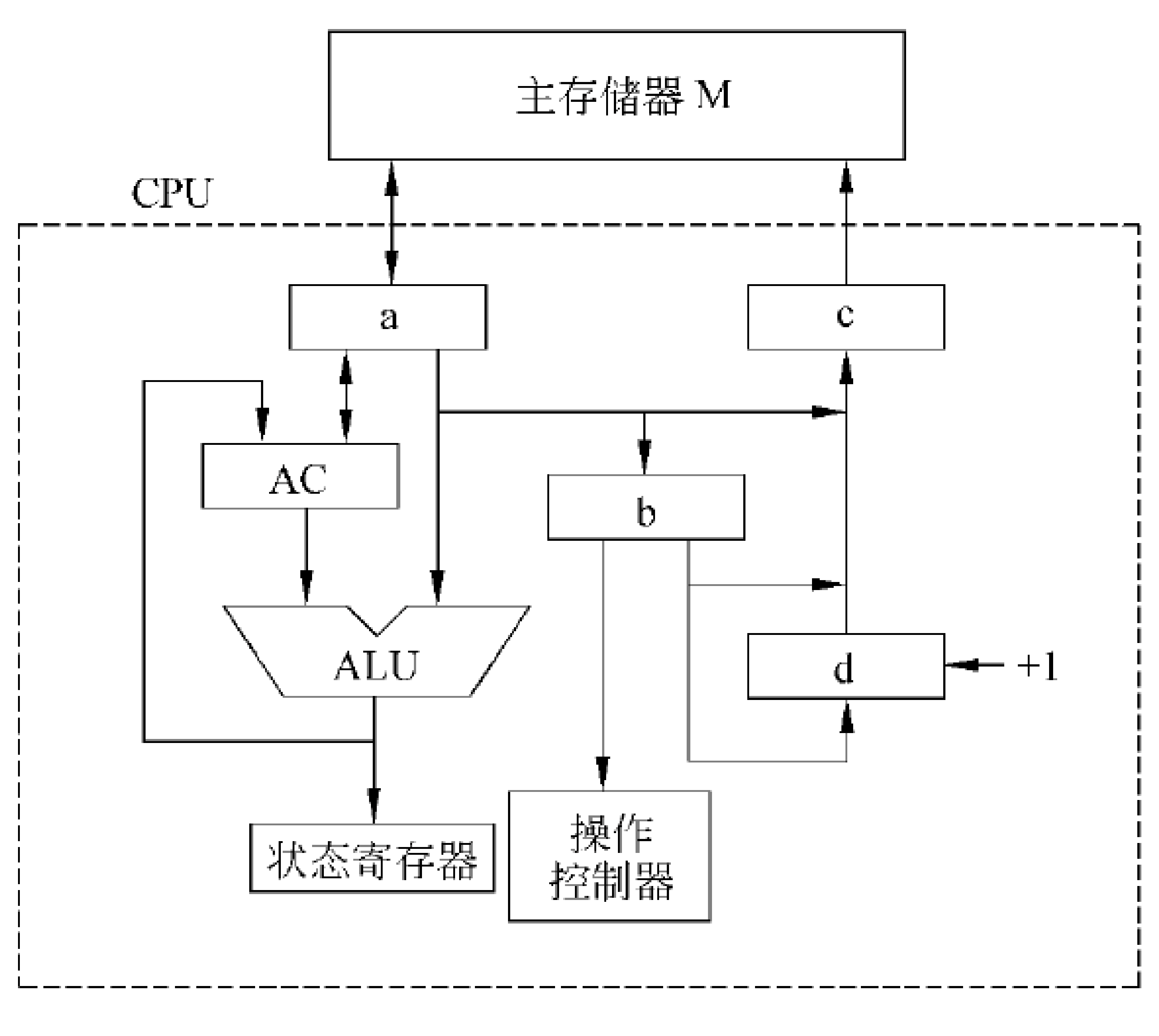


图 6-7　某机 CPU 结构

(3) 简述完成指令 LDA X 的数据通路(X 为主存地址,LDA 的功能为(X)→AC)。

(4) 简述完成指令 ADD Y 的数据通路(Y 为主存地址,ADD 的功能为(AC)+(Y)→AC)。

(5) 简述完成指令 STA Z 的数据通路(Z 为主存地址,STA 的功能为(AC)→Z)。

解:(1) 这 4 个寄存器中,a 为存储器数据寄存器 MDR,b 为指令寄存器 IR,c 为存储器地址寄存器 MAR,d 为程序计数器 PC。

(2) 取指令的数据通路:

PC→MAR→MM→MDR→IR。

(3) 指令 LDA X 的数据通路:

X→MAR→MM→MDR→ALU→AC。

(4) 指令 ADD Y 的数据通路:

Y→MAR→MM→MDR↘
　　　　　　　　　AC↗ALU→AC。

(5) 指令 STA Z 的数据通路:

Y→MAR,AC→MDR→MM。

【例 6.4】 某计算机的 CPU 内部为双总线结构,所有数据传送都通过 ALU,ALU 还具有下列功能,CPU 见图 6-8 结构。

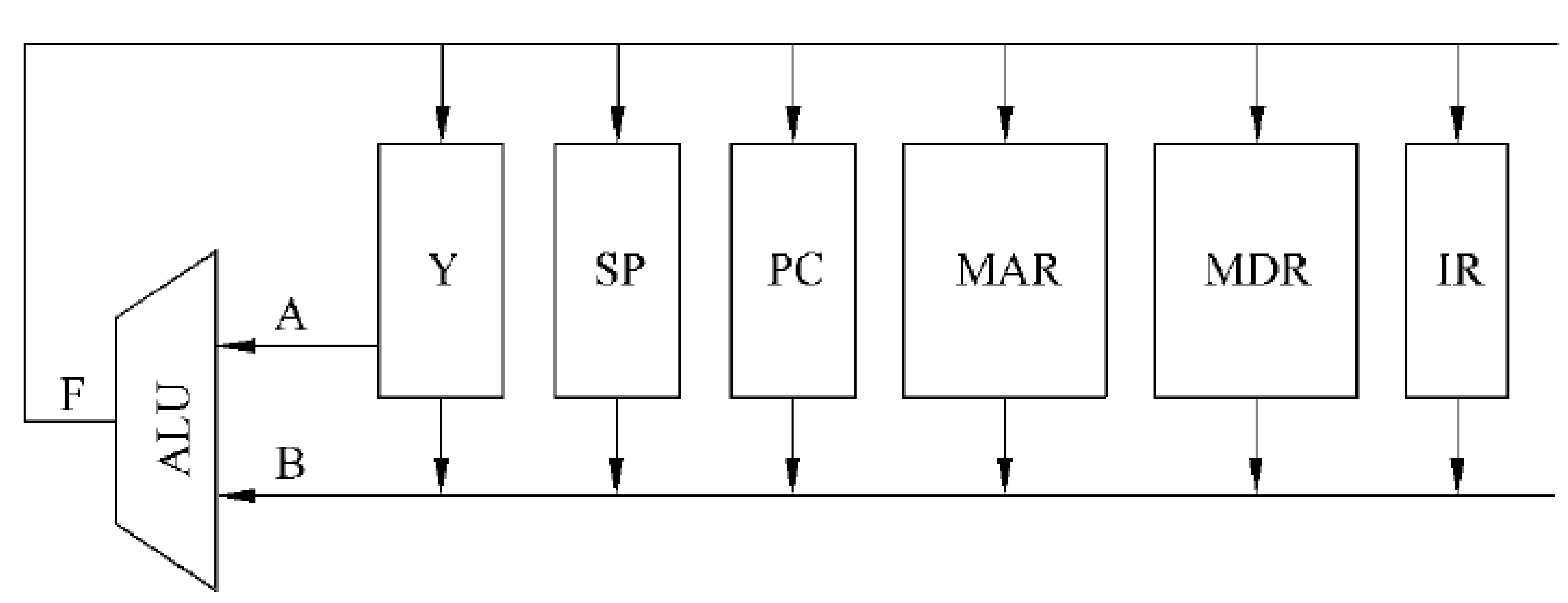


图 6-8　CPU 内部结构

F=A;　　　F=B
F=A+1;　　F=B+1
F=A-1;　　F=B-1

写出转子指令(JSR)的取指和执行周期的微操作序列。JSR 指令占两个字,第一个字是操作码,第二个为子程序的入口地址。返回地址保存在存储器堆栈中,堆栈指针始终指向栈顶。图 6-8 中 Y 为暂存器,PC 为程序计数器,MAR 和 MDR 分别为存储器地址和数据寄存器,IR 为指令寄存器。

解:转子指令由两个字组成,第一个字为操作码,第二个字为子程序的入口地址。这条指令的微操作序列如下:

① PC→B,F=B,F→MAR,Read　　;取指令的第一个字
② PC→B,F=B+1,F→PC
③ MDR→B,F=B,F→IR
④ PC→B,F=B,F→MAR,Read　　;取指令的第二个字
⑤ PC→B,F=B+1,F→PC
⑥ MDR→B,F=B,F→Y
⑦ SP→B,F=B-1,F→SP,F→MAR　　;修改栈指针,返回地址压入堆栈
⑧ PC→B,F=B,F→MDR,Write
⑨ Y→A,F=A,F→PC　　;子程序的首地址→PC
⑩ End

【例 6.5】 设 CPU 内部结构如图 6-9 所示,此外还设有 B、C、D、E、H、L 6 个寄存器,它们各自的输入和输出端都与内部总线相通,并分别受控制信号控制(如 B_{in} 为寄存器 B 的输入控制;B_{out} 为寄存器 B 的输出控制),假设 ALU 的结果直接送入 Z 寄存器中。要求从取指令开始,写出完成下列指令所需的控制信号。

ADD　B,C　　(B)+(C)→B
SUB　A,H　　(AC)-(H)→AC

解:两条指令的微操作序列如下。

ADD B,C 指令:

① PC_{out},MAR_{in},Read　　;(PC)→MAR,Read
②+1,MDR_{out},IR_{in}　　;(PC)+1→PC,M(MAR)→MDR→IR
③ B_{out},Y_{in}
④ C_{out},ALU_{in},"+"　　;(B)+(C)→Z
⑤ Z_{out},B_{in}　　;(Z)→B

SUB A,H 指令:

① PC_{out},MAR_{in},Read　　;(PC)→MAR,Read
②+1,MDR_{out},IR_{in}　　;(PC)+1→PC,M(MAR)→MDR→IR
③ AC_{out},Y_{in}
④ H_{out},ALU_{in},"-"　　;(AC)-(H)→Z
⑤ Z_{out},AC_{in}　　;(Z)→AC

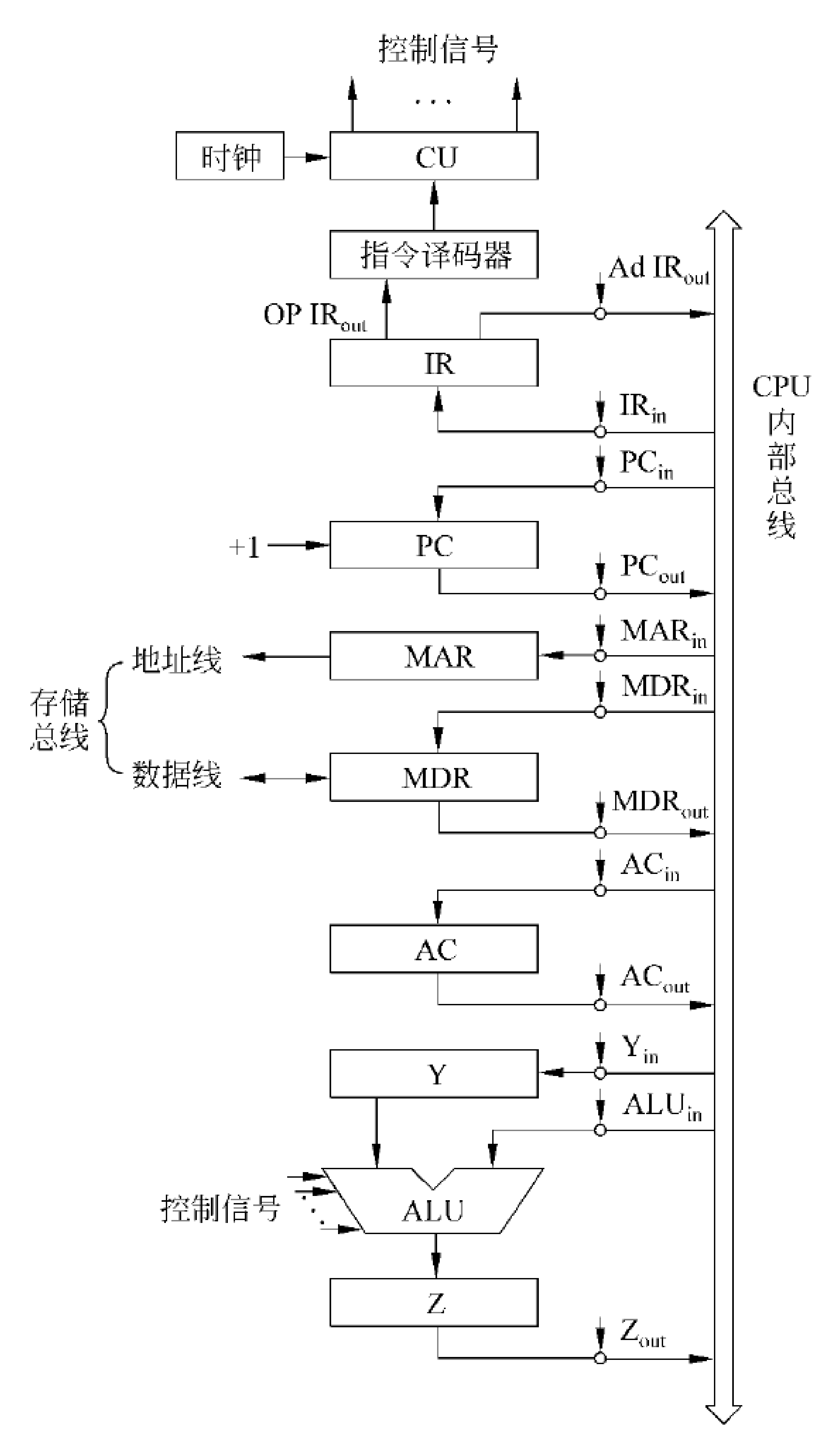


图 6-9　某机 CPU 内部结构

【例 6.6】 某机主要部件如图 6-10 所示。

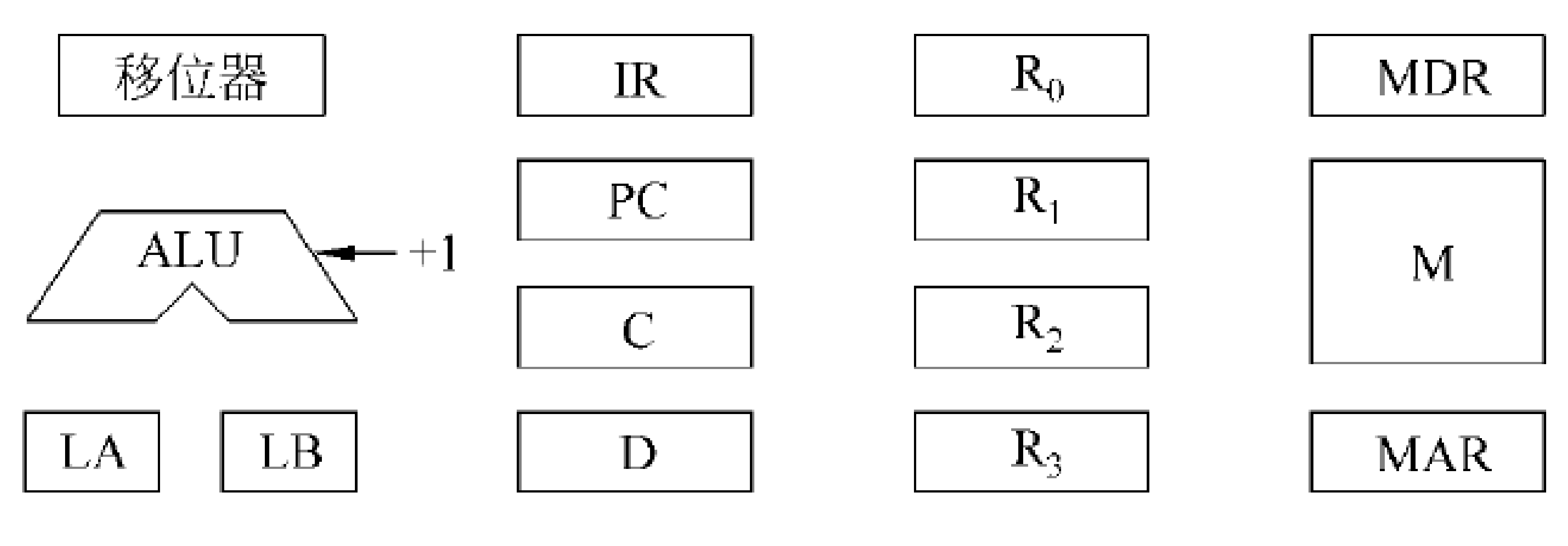


图 6-10　某机主要部件

(1) 请补充各部件间的主要连接线，并注明数据流动方向。

(2) 拟出指令 ADD (R_1)，$(R_2)+$ 的执行流程（含取指过程与确定后继指令地址）。该指令的含义是进行加法操作，源操作数地址和目的操作数地址分别在寄存器 R_1 和 R_2 中，目的操作数寻址方式为自增型寄存器间址。

解：(1) 将各部件间的主要连接线补充完后如图 6-11 所示。

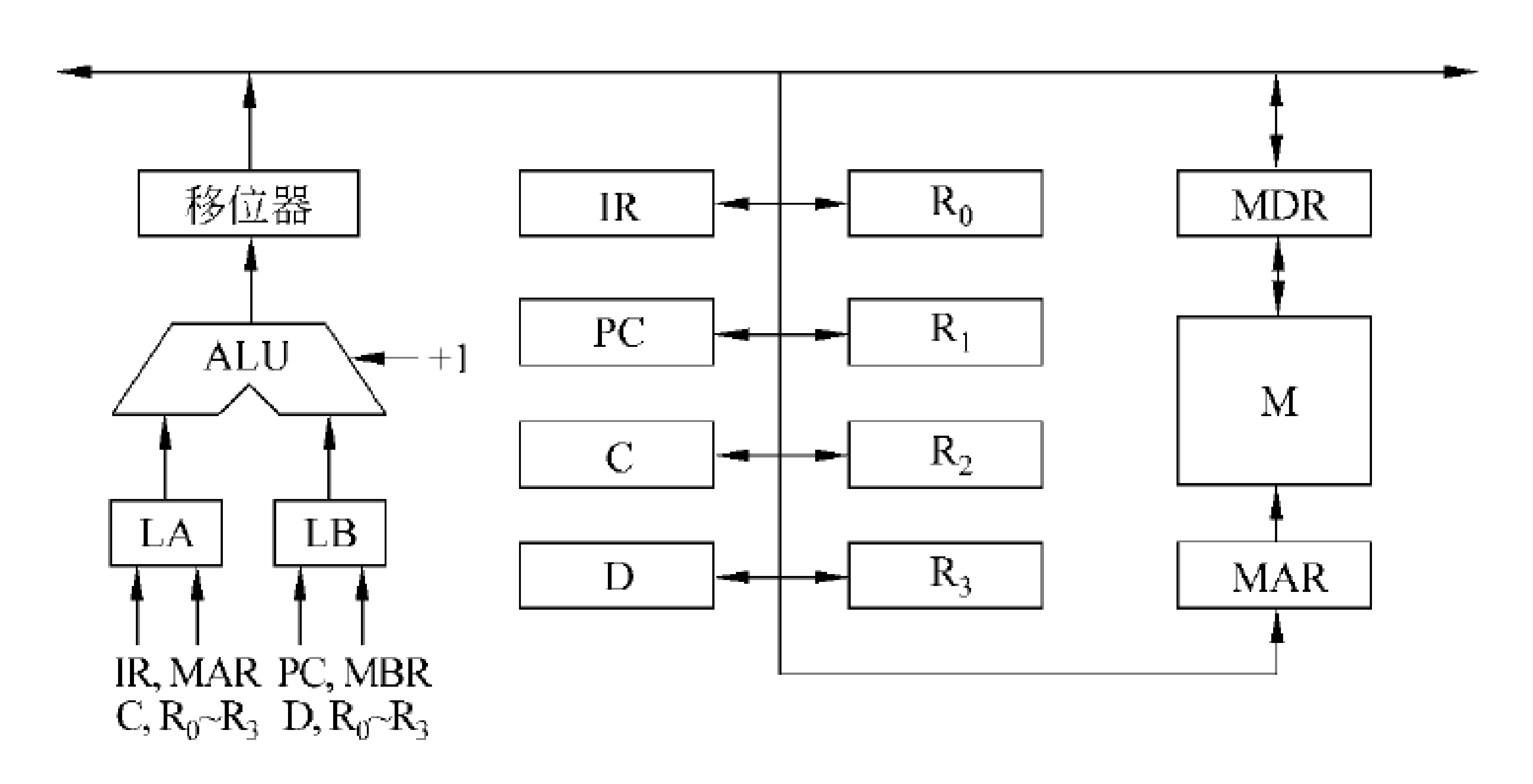


图 6-11　某机数据通路

(2) 指令 ADD (R_1),(R_2)+的含义为:

((R_1))+((R_2))→(R_2)

(R_2)+1→R_2

指令的执行流程如下:

① (PC)→MAR　　;取指令
② Read
③ M(MAR)→MDR→IR
④ (PC)+1→PC
⑤ (R_1)→MAR　　;取被加数
⑥ Read
⑦ M(MAR)→MDR→C
⑧ (R_2)→MAR　　;取加数
⑨ Read
⑩ M(MAR)→MDR→D
⑪ (R_2)+1→R_2　　;修改目的地址
⑫ (C)+(D)→MDR　　;求和并保存结果
⑬ Write
⑭ MDR→MM

【例 6.7】 设 CPU 中各部件及其相互连接关系如图 6-12 所示。图中 W 是写控制标志,R 是读控制标志,R_1 和 R_2 是暂存器。

(1) 假设要求在取指周期由 ALU 完成 (PC)+1→PC 的操作(即 ALU 可以对它的一个源操作数完成加 1 的运算)。要求以最少的节拍写出取指周期全部微操作控制信号及节拍安排。

(2) 写出指令 ADD #α(#为立即寻址特征,隐含的操作数在 ACC 中)在执行阶段所需的微操作控制信号及节拍安排。

解:(1) 由于程序计数器 PC 本身没有计数功能,(PC)+1→PC 需由 ALU 完成,因此 PC 的值可作为 ALU 的一个源操作数,靠控制 ALU 做+1 运算得到 (PC)+1,结果送至与 ALU 输出端相连的 R_2,然后再送至 PC。上述微操作控制信号中 T_1 节拍的(PC)+1→R_2 和 T_3 节拍的 R_2→PC 共同完成(PC)+1→PC 的功能。T_2 节拍中的 IR_{OP} 代表指令的操作

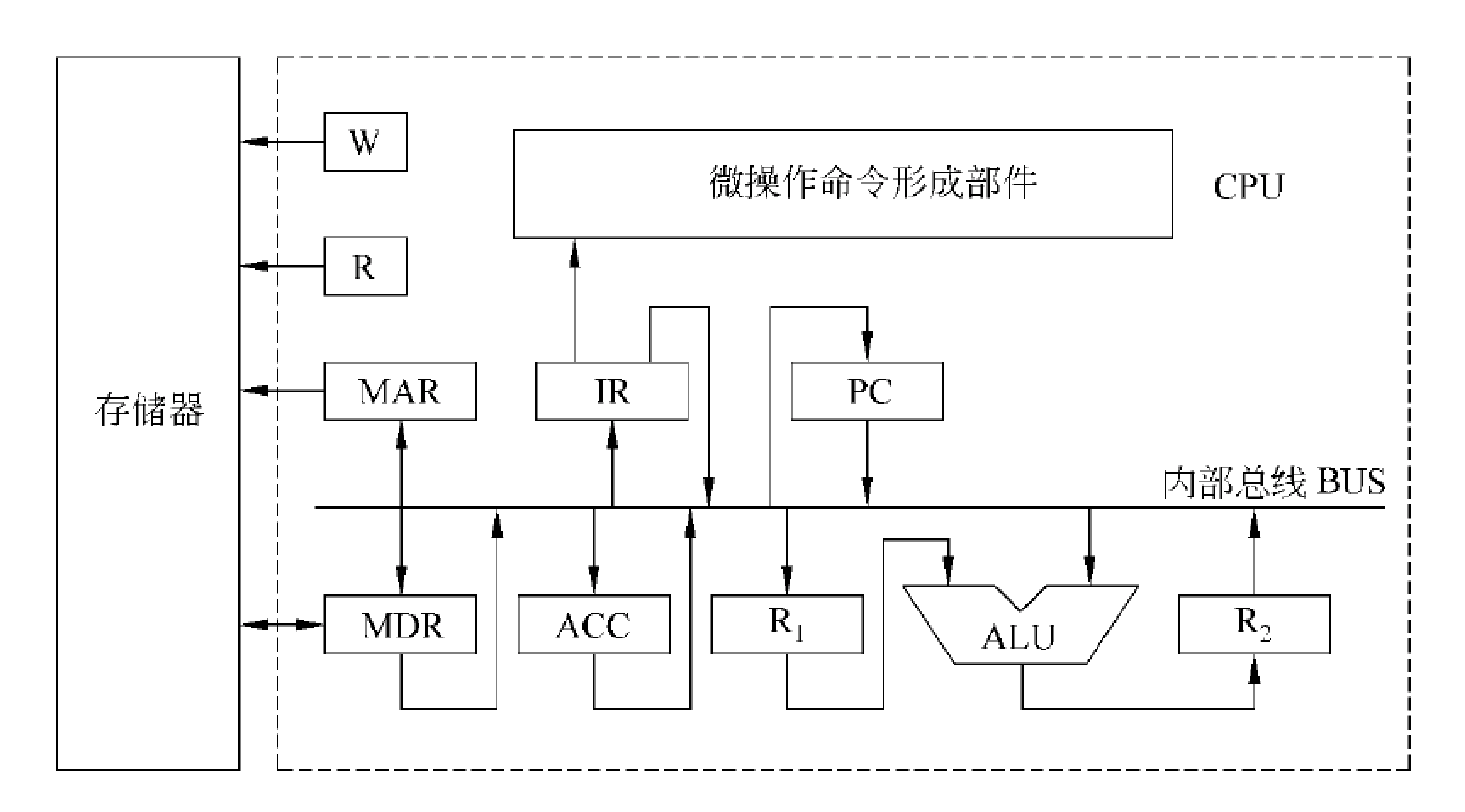


图 6-12　例 6.7 的 CPU 结构

码字段。取指周期的微操作控制信号及节拍安排如下：

T_0　PC→MAR, 1→R

T_1　M(MAR)→MDR, (PC)+1→R_2

T_2　MDR→IR, IR_{OP}→微操作命令形成部件

T_3　R_2→PC

(2) 由于指令的地址码字段中存放的就是立即数，所以首先将立即数送至 R_1，然后将隐含在 ACC 中的另一操作数与立即数相加，结果送回 ACC。T_0 节拍中 IR_{addr} 代表指令的地址码字段。立即寻址的加法指令执行周期的微操作控制信号及节拍安排如下：

T_0	IR_{addr}→R_1	;立即数→R_1
T_1	(R_1)+(ACC)→R_2	;ACC 的内容通过总线送 ALU，与立即数相加
T_2	R_2→ACC	;结果→ACC

【例 6.8】 已知带返转指令的含义如图 6-13 所示，写出机器在完成带返转指令时，取指阶段和执行阶段所需的全部微操作控制信号及节拍安排。

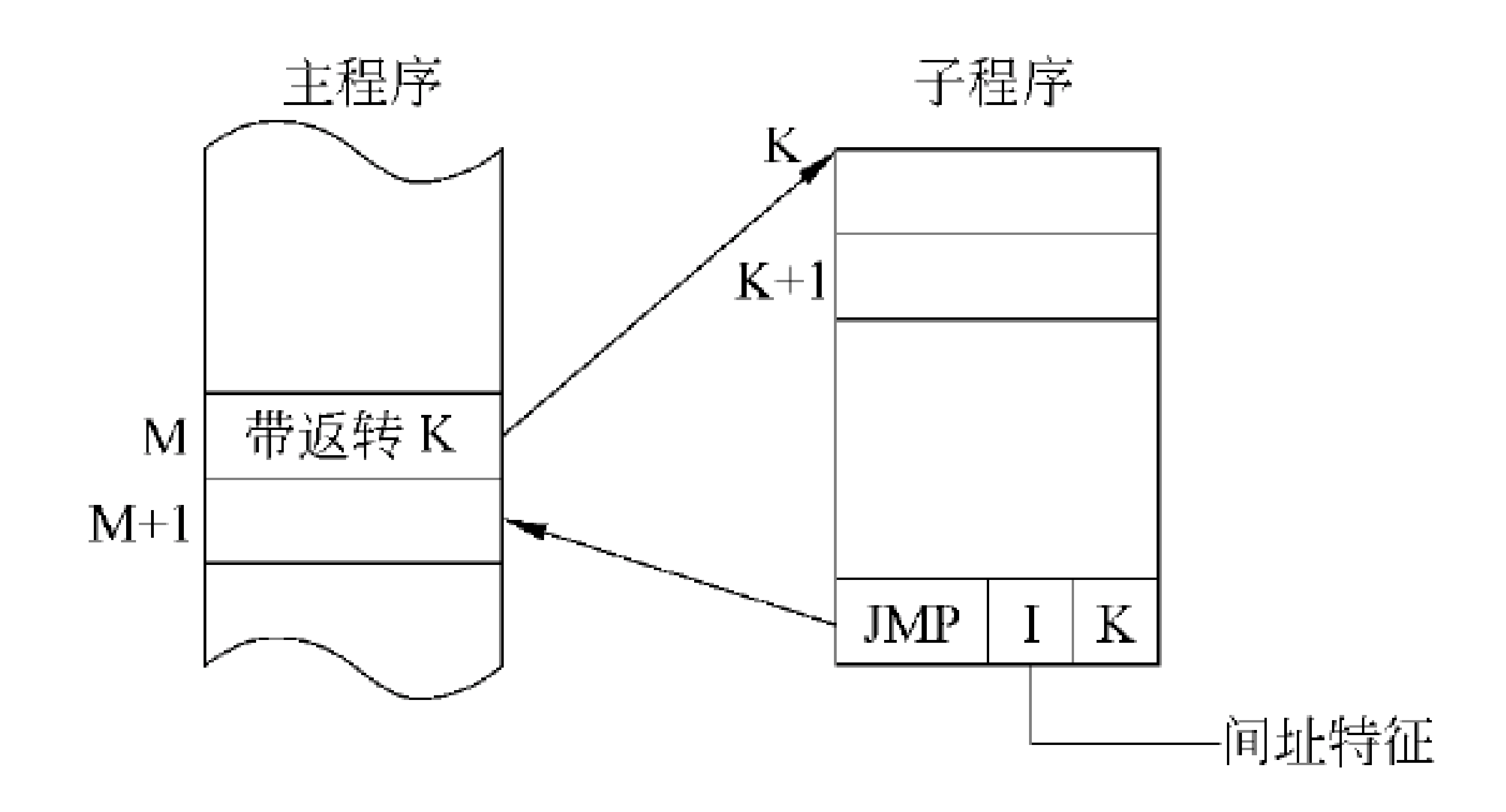


图 6-13　带返转指令执行示意图

解：带返转指令是一种特殊的转子指令，在执行这条指令时，将主程序的返回地址自动地存入子程序的第一个单元，以便当子程序返回时，采用间接寻址的无条件转移指令返回主程序。取指阶段是公操作，与其他指令无异，不需多加分析。关键在执行阶段，由图 6-13 可见，执行阶段需完成将返回地址（M＋1），存入子程序首地址单元（K）中，然后将真正的子程

序入口地址(K+1)送给PC。取指阶段的微操作控制信号及节拍安排如下：

T_0　PC→MAR,1→R
T_1　M(MAR)→MDR,PC+1→PC
T_2　MDR→IR

执行阶段的T_0节拍，指令的地址码字段(K)送到存储器地址寄存器；T_1节拍，将返回地址(M+1)送存储器数据寄存器，然后发出写命令；T_2节拍，返回地址被写入K号单元中，并产生真正的子程序入口地址(K+1)。执行阶段的微操作控制信号及节拍安排如下：

T_0　IR_{addr}→MAR
T_1　PC→MDR,1→W
T_2　MDR→M(MAR),Ad(IR)+1→PC

【例6.9】 单总线CPU结构以及数据通路如图6-14所示，其中MAR为地址寄存器，MDR为数据寄存器，MEM为主存储器，R_0～R_3为通用寄存器，PSW为状态寄存器，Y、Z为暂存寄存器，PC为程序计数器，IR为指令寄存器。

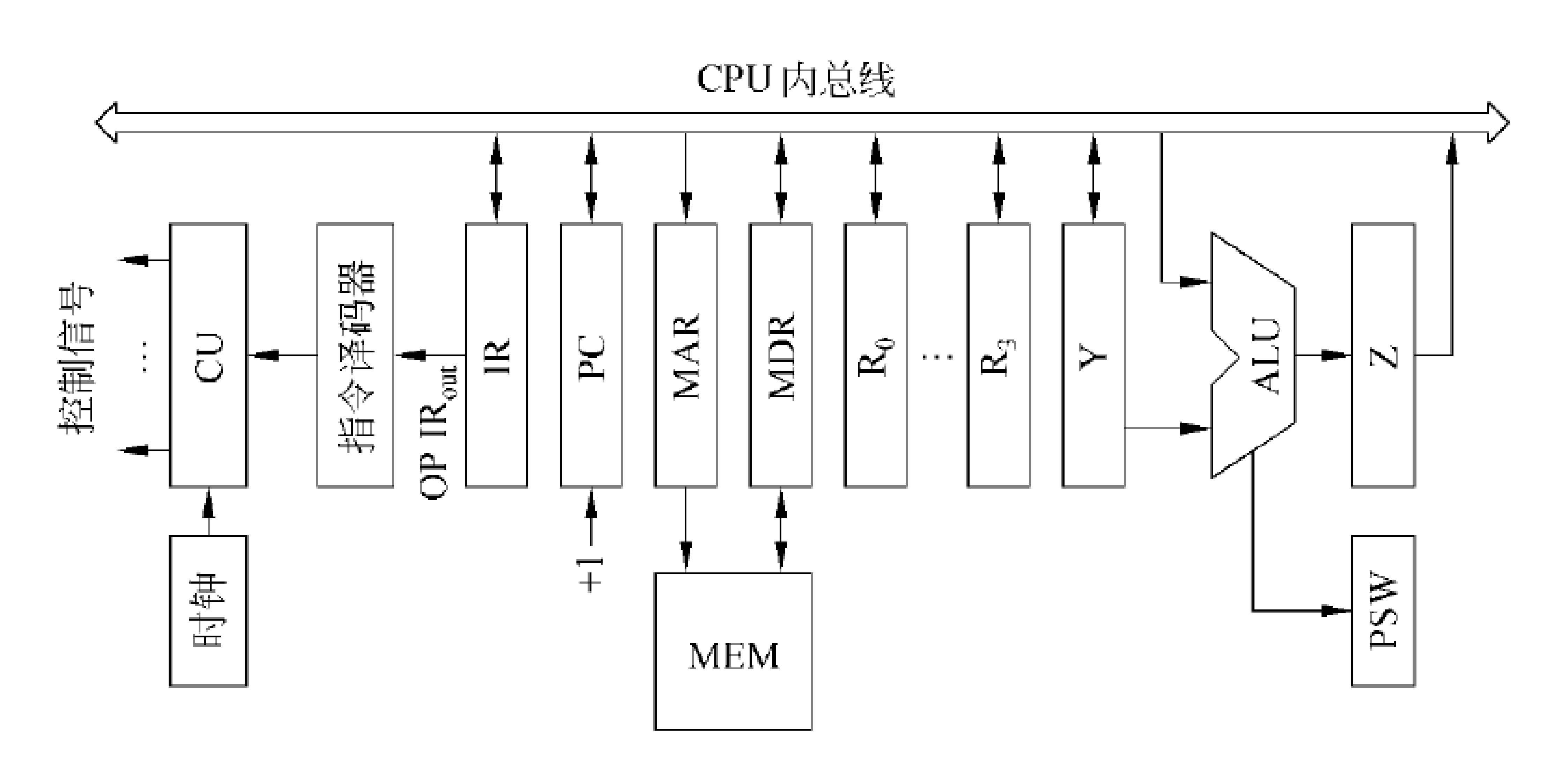


图6-14　某单总线CPU结构

请用寄存器级传送形式设计下列指令执行的分步流程(包括取指令阶段)。

(1) AND R_0,Addr。

(2) ADD R_0,Offs(R_1)。

解：(1) AND R_0,Addr的含义是，将以Addr为直接地址的存储单元内容读出，假设指令占2个字节，Addr是指令的第2字节。Addr的内容与R_0内容进行逻辑乘运算，结果存入R_0中。

① PC→MAR,MAR→ABUS,Read　　;送指令地址(第1字节地址)到地址寄存器
② DBUS→MDR,PC+1→PC
③ MDR→IR　　;取指令(第1字节)到指令寄存器
④ PC→MAR,MAR→ABUS,Read　　;送下一地址(第2字节地址)到地址寄存器
⑤ DBUS→MDR,PC+1→PC
⑥ MDR→MAR,MAR→ABUS,Read　　;取出直接地址送地址寄存器
⑦ DBUS→MDR,MDR→Y　　;取操作数
⑧ R_0∧Y→Z　　;两数相与
⑨ Z→R_0　　;结果送R_0

(2) 在 ADD R_0,Offs(R_1)指令中,R_0 为目的地址,采用寄存器寻址,Offs(R_1)为源地址,采用变址寻址,Offs 代表形式地址,R_1 代表变址寄存器。假设指令占 2 个字节,Offs 是指令的第 2 字节。

①	PC→MAR,MAR→ABUS,Read	;送指令地址(第 1 字节地址)到地址寄存器
②	DBUS→MDR,PC+1→PC	
③	MDR→IR	;取指令(第 1 字节)到指令寄存器
④	PC→MAR,MAR→ABUS,Read	;送下一地址(第 2 字节地址)到地址寄存器
⑤	DBUS→MDR,PC+1→PC	
⑥	MDR→Y	;取出形式地址
⑦	R_1+Y→Z	;变址值和形式地址相加,求得有效地址
⑧	Z→MAR,MAR→ABUS,Read	;送有效地址到地址寄存器
⑨	DBUS→MDR	
⑩	MDR→Y	;取出源操作数
⑪	R_0+Y→Z	;两数相加
⑫	Z→R_0	;结果送 R_0

【例 6.10】 已知某机采用微程序控制方式,其控制存储器的容量为 512×48 位。微程序可在整个控制存储器中实现转移,可控制微程序转移的条件共有 4 个(直接控制),微程序采用水平型格式,如图 6-15 所示。

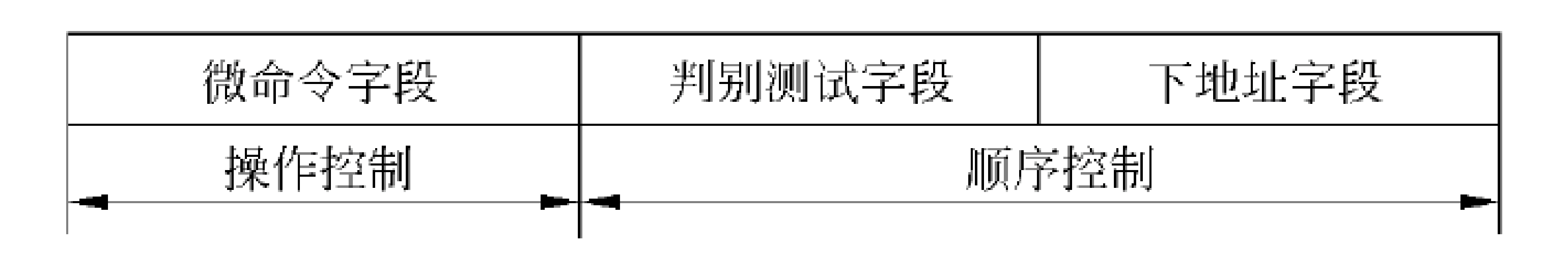


图 6-15 某机微指令格式

(1) 微指令中的 3 个字段分别应为多少位?为什么?

(2) 画出围绕这种微指令格式的微程序控制器逻辑框图。

解:(1) 因为控制微程序转移的条件采用直接控制,即每一位对应一个转移条件,故判别测试字段为 4 位。因为控存容量为 512 个单元,所以下地址字段为 9 位。微命令字段则是(48－4－9)＝35 位。

(2) 对应上述微指令格式的微程序控制器逻辑框图如图 6-16 所示。

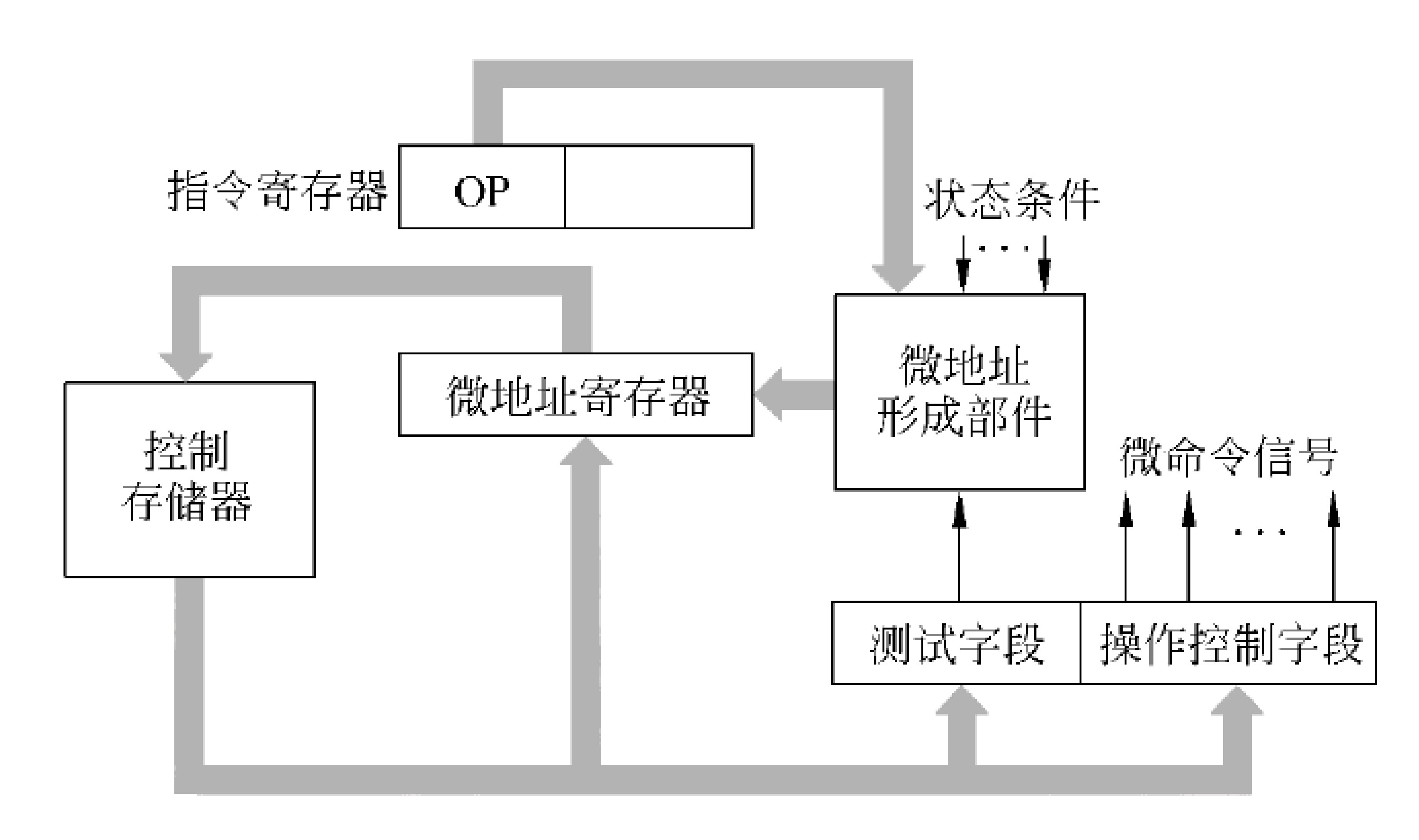


图 6-16 微程序控制器逻辑框图

图6-16中微地址形成部件的输入是指令寄存器的操作码，各种状态条件以及判断测试字段所给的判别标志(某一位为1)，其输出修改微地址寄存器的适当位数，从而实现微程序的分支转移。

【例6.11】 某机采用微程序控制器设计，已知每条机器指令的执行过程均可分解成8条微指令组成的微程序，该机指令系统采用6位定长操作码格式，控制存储器至少应能容纳多少条微指令？如何确定机器指令操作码与该指令微程序的入口地址的对应关系，请给出具体方案。

解：由于每条机器指令都可以分解为8条微指令，并且机器指令系统采用6位定长操作码，总共允许有2^6种不同的机器指令，控制存储器可容纳的微指令条数为$64\times8=512$。

由于控存的容量为512个单元，所以微地址寄存器为9位，其中高6位为机器指令的操作码，它与任意的低3位拼接即可形成微程序的入口地址，如图6-17所示。相邻两条机器指令的微程序入口地址相差8个单元。

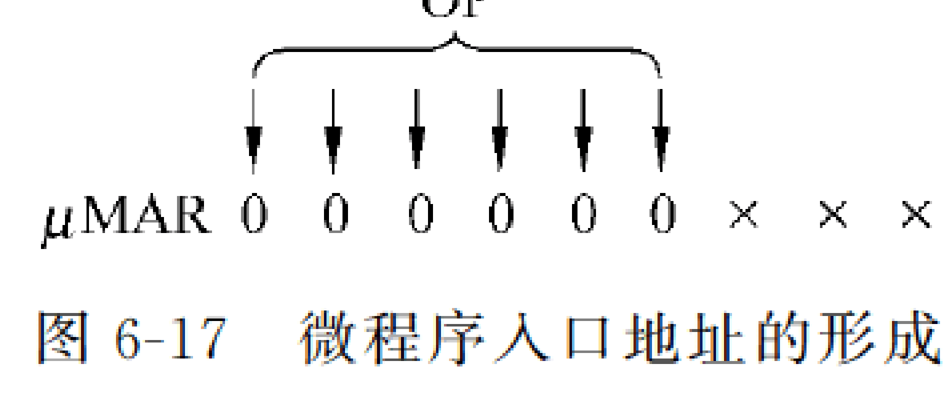


图6-17　微程序入口地址的形成

【例6.12】 图6-18为一微程序流程，每一方框代表一条微指令，分别用字符A～P表示其执行的微操作，根据给定的微程序流程设计微指令的顺序控制字段，并为每条微指令分配一个微地址。

解：该微程序流程有两处有分支的地方，第一处有4个分支，由指令操作码的I_1I_0 2位指向4条不同的微指令，第二处有2个分支，根据运算结果Z的值决定后继微地址，因此微指令顺序控制部分中的测试字段应为2位。因为共有16条微指令，则下址字段用4位，微指令的格式如图6-19所示。

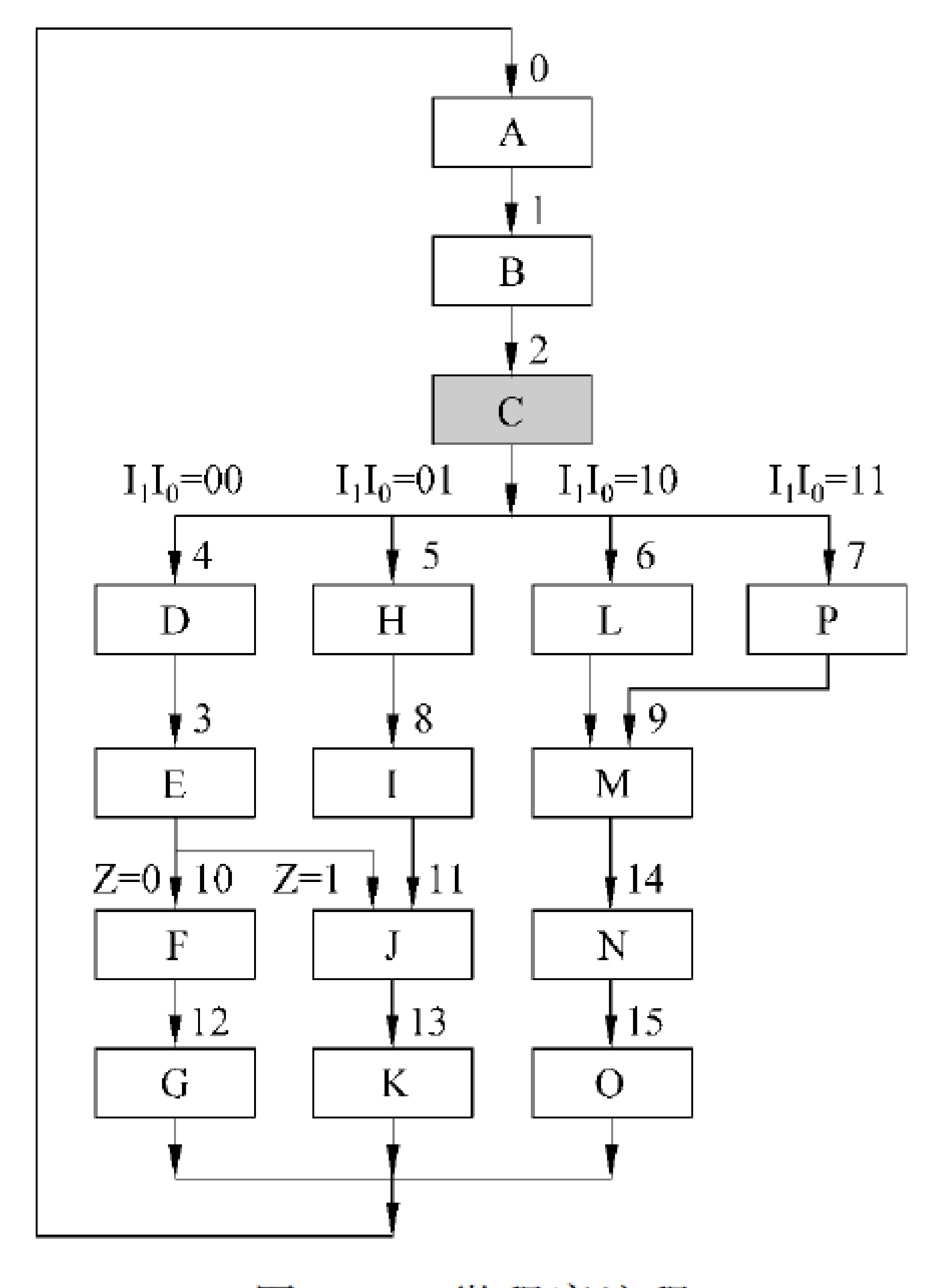


图6-18　微程序流程

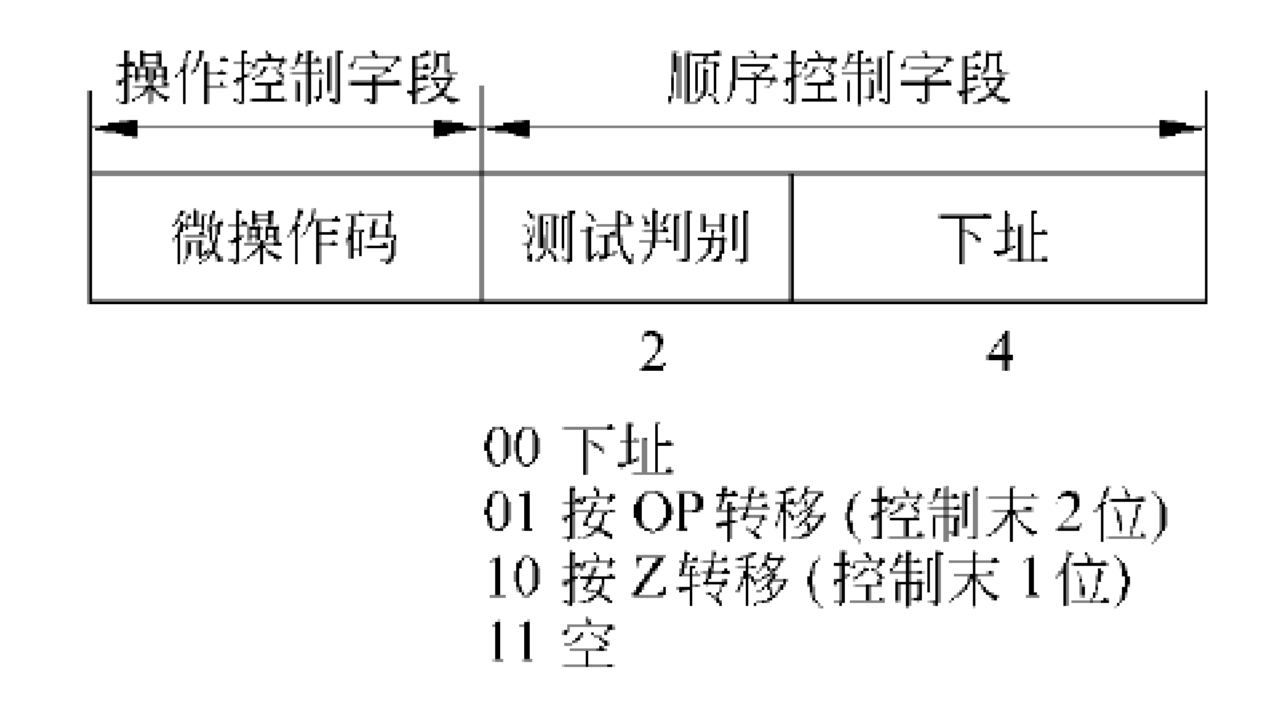


图6-19　微指令格式

微地址分配的关键是带有分支的微指令，此时，下址字段的值具有一定的约束条件，一般要选择测试条件控制的那几位为全0，目的是简化地址修改逻辑。如本例中的第3条微

指令(图 6-18 中阴影框)，由于该微指令按指令操作码 I_1I_0 实现 4 路分支，它控制在末 2 位，因此，第 3 条微指令的下址约束条件是末 2 位全为 0。这里选择为 0100，它的后继 4 条微指令的地址分别为 0100、0101、0110、0111，末 2 位就是 I_1I_0 的值。按 Z 实现 2 路分支的情况与此类似。

余下的微指令地址没有约束条件，可以任意分配。一般根据微程序流程从小到大，把控存中没有分配的微地址分配到不同的微指令中，就得到全部微指令地址。

微程序流程对应的微地址和微指令如表 6-3 所示。

表 6-3　微程序流程对应的微地址和微指令

微地址		微指令		
		操作控制字段	顺序控制字段	
二进制	十进制	微命令	测试判别	下地址
0000	0	A	00	0001
0001	1	B	00	0010
0010	2	C	01	0100
0011	3	E	10	1010
0100	4	D	00	0011
0101	5	H	00	1000
0110	6	L	00	1001
0111	7	P	00	1001
1000	8	I	00	1011
1001	9	M	00	1110
1010	10	F	00	1100
1011	11	J	00	1101
1100	12	G	00	0000
1101	13	K	00	0000
1110	14	N	00	1111
1111	15	O	00	0000

微地址修改电路如图 6-20 所示。

【例 6.13】　已知某运算器的基本结构如图 6-21 所示，它具有+(加)、-(减)、M(传送)3 种操作。

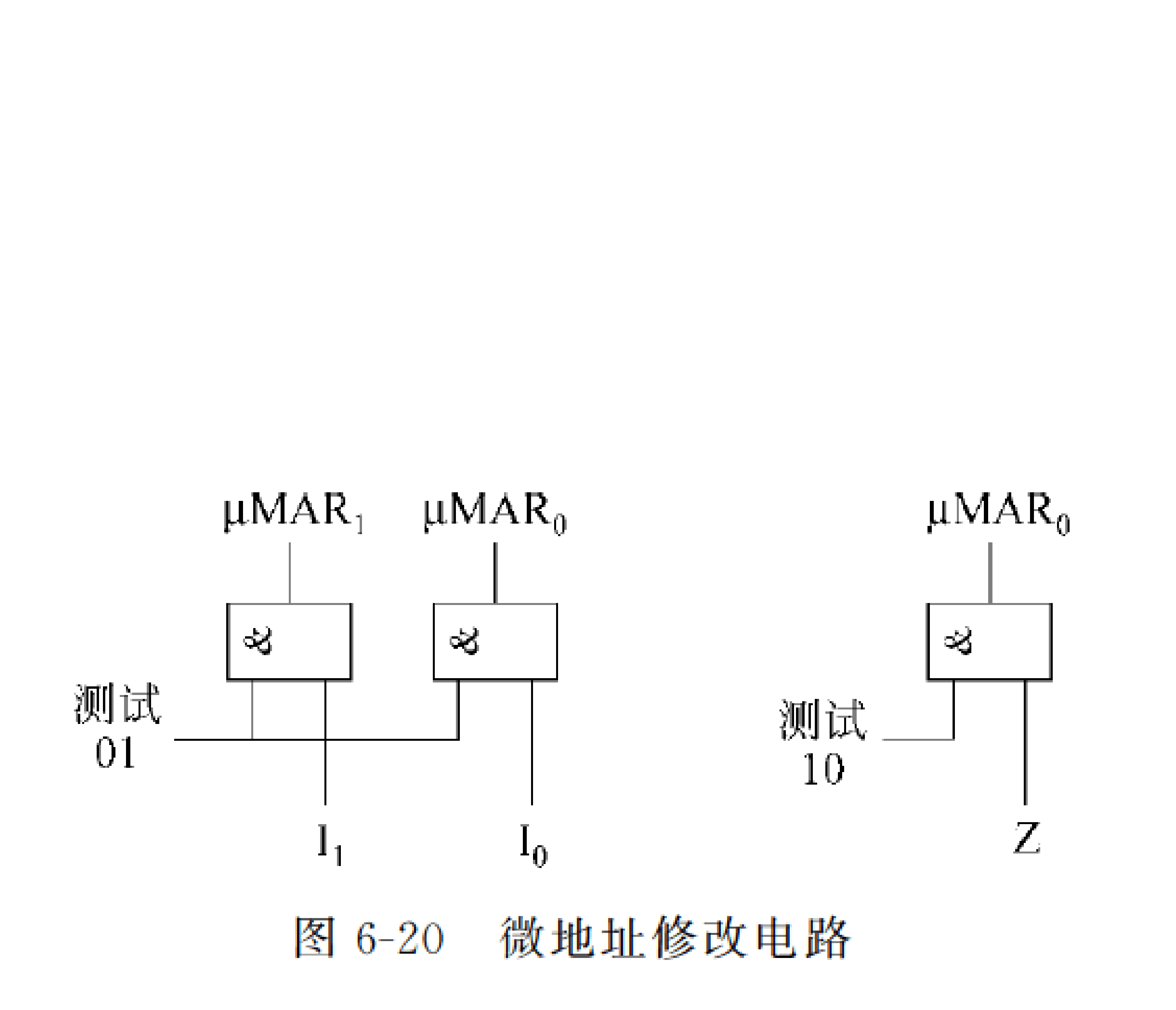


图 6-20　微地址修改电路

图 6-21　某运算器的基本结构

(1) 写出图6-21中1～12表示的运算器操作的微命令。

(2) 指出互斥性微命令。

(3) 设计适合此运算器的微指令格式。

(4) 指令DDA的功能是计算R_1、R_2和R_3这3个寄存器的和,若进位C=0,则$R_1+R_2 \to R_2$;若进位C=1,则$R_1+R_2+R_3 \to R_2$,请画出指令DDA的微程序流程图。

(5) 按(3)设计的微指令格式将DDA的微程序代码化。

解:(1) 图6-21中1～12表示的运算器操作的微命令分别为:

1: +	2: -	3: M
4: $R_1 \to A$	5: $R_2 \to A$	6: $R_3 \to A$
7: $R_1 \to B$	8: $R_2 \to B$	9: $R_3 \to B$
10: $BUS \to R_1$	11: $BUS \to R_2$	12: $BUS \to R_3$

(2) 根据图6-21的数据通路,以下3组微命令是互斥的:

① +、-、M

② $R_1 \to A$、$R_2 \to A$、$R_3 \to A$

③ $R_1 \to B$、$R_2 \to B$、$R_3 \to B$

最后3个微命令:$BUS \to R_1$,$BUS \to R_2$,$BUS \to R_3$,从数据通路来看是兼容的,但从操作上来看是互斥的,所以还是将它们放在一个字段中。

(3) 适合此运算器的微指令格式如图6-22所示。其操作控制字段8位,分为4个字段,各个字段的安排如下:

字段1	字段2	字段3	字段4
00: 不操作	00: 不操作	00: 不操作	00: 不操作
01: +	01: $R_1 \to A$	01: $R_1 \to B$	01: $BUS \to R_1$
10: -	10: $R_2 \to A$	10: $R_2 \to B$	10: $BUS \to R_2$
11: M	11: $R_3 \to A$	11: $R_3 \to B$	11: $BUS \to R_3$

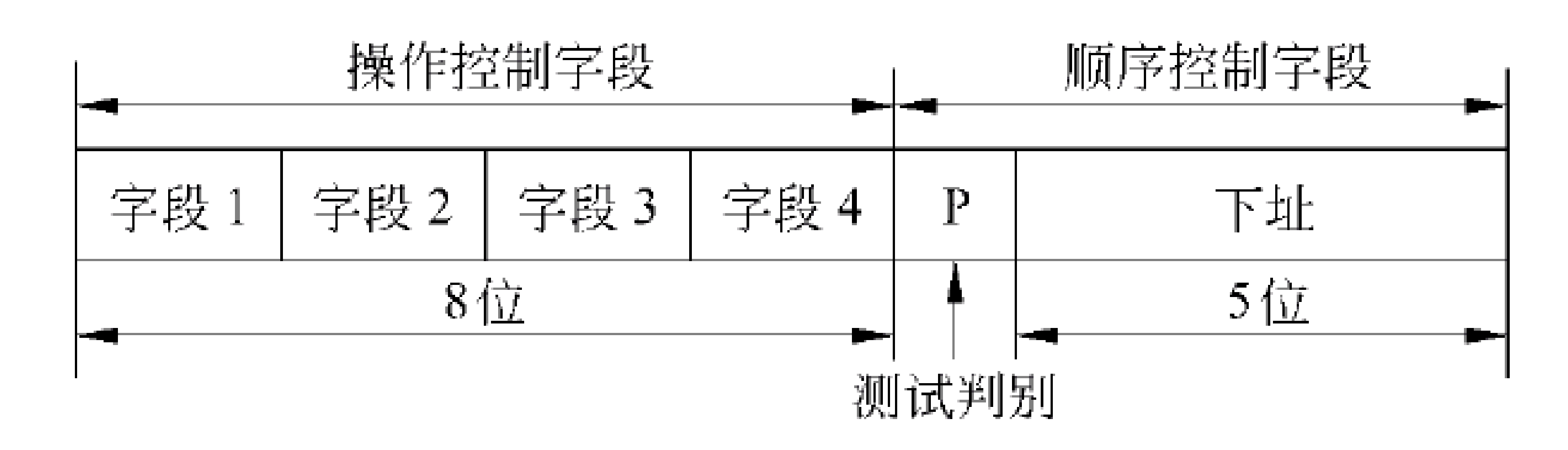


图6-22 例6.13的微指令格式

(4) 指令DDA的微程序流程图如图6-23所示。

(5) 设下址地址为5位,控制字段为1位,P=0时顺序控制;P=1时由C修改微地址的μMAR_3和μMAR_1,如图6-24所示。

假设各微指令的微地址安排如图6-23所示,DDA指令的微程序代码如表6-4所示。

取指微指令在00000单元,在表6-3中没有标出。

DDA指令的第一条微指令放在01000单元,01000单元执行后按下址地址转01001单元;第二条微指令在01001单元,01001单元中的测试判别位P=1,下址地址为00000,当C=0时,下一条微指令的地址为00000,即开始下一条机器指令的取指微指令;当C=1时,

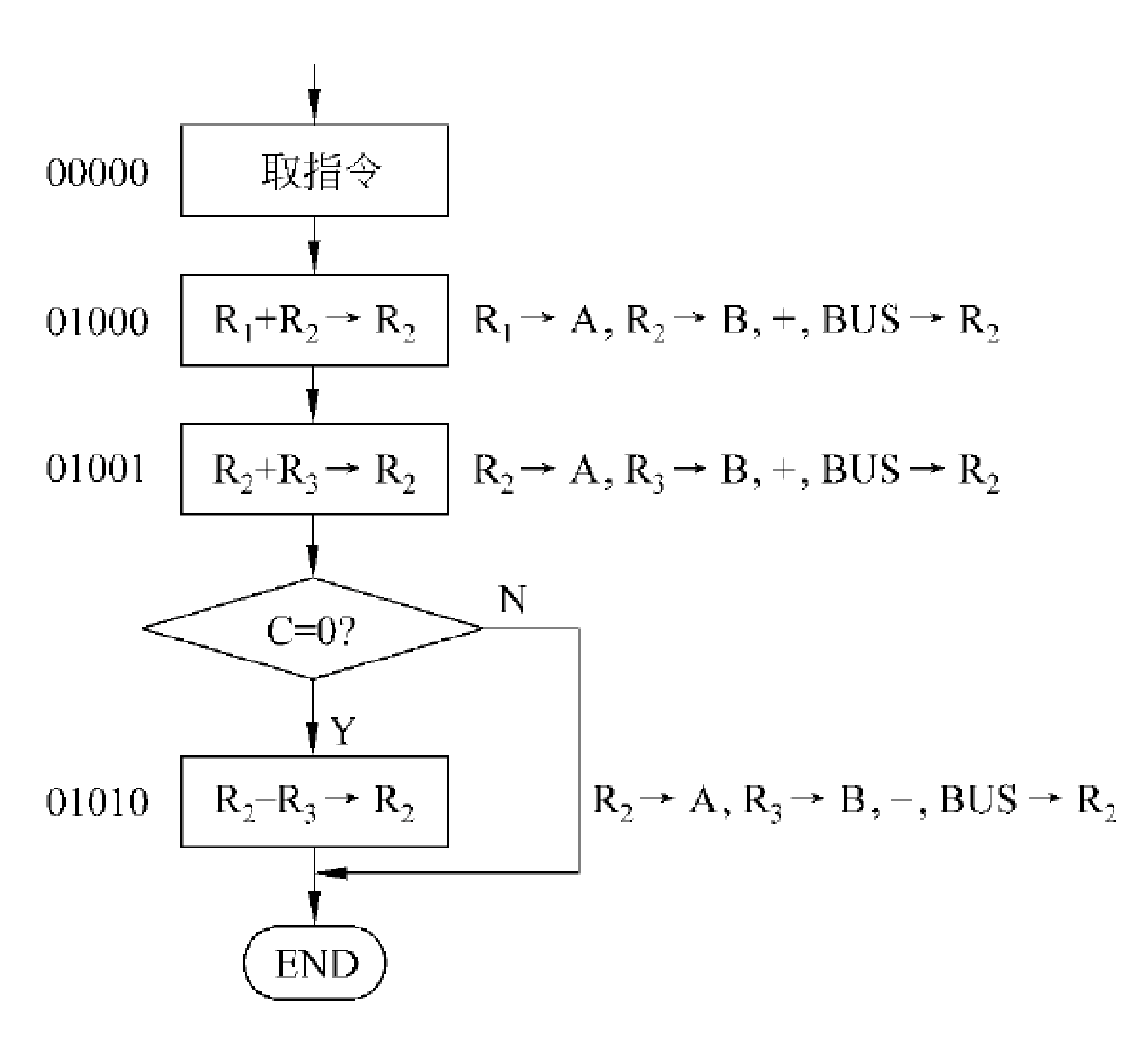


图 6-23　微程序流程图

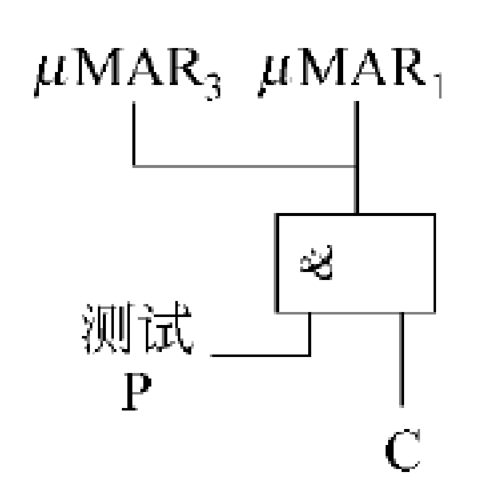


图 6-24　微地址的 μMAR_3 和 μMAR_1 的形成

表 6-4　DDA 指令的微程序代码

微　地　址	微　指　令
01000	01　01　10　10　0　01001
01001	01　10　11　10　1　00000
01010	10　10　11　10　0　00000

将 μMAR_3 和 μMAR_1 修改为 1，形成下一条微指令的地址 01010；01010 单元执行完后按下址地址转 00000 单元，即开始下一条机器指令的取指微指令。

【例 6.14】　某机有 8 条微指令 $I_1\sim I_8$，每条微指令所含的微命令控制信号如表 6-5 所示。

表 6-5　微指令所含的微命令控制信号

微指令	微命令信号									
	a	b	c	d	e	f	g	h	i	j
I_1	√	√	√	√	√					
I_2	√			√		√	√			
I_3		√						√		
I_4			√							
I_5			√		√		√		√	
I_6	√							√		√
I_7			√	√				√		
I_8	√	√						√		

a～j 分别代表 10 种不同性质的微命令信号，假设一条微指令的操作控制字段为 8 位，

请安排微指令的操作控制字段格式,并将全部微指令代码化。

解:本系统中有10种不同性质的微命令信号,但一条微指令的操作控制字段只有8位,所以不能采用直接控制法。又因为微指令中有多个微命令是兼容的微命令,必须同时出现,如微指令I_1中的微命令a~e,故也不能采用最短编码法。

最终选用字段编码法和直接控制法相结合的方法。将互斥的微命令安排在同一段内,兼容的微命令安排在不同的段内。b、i、j 3个微命令是互斥的微命令,把它们安排在一个段内;e、f、h 3个微命令也是互斥的,把它们也安排在另一个段内。此微指令的操作控制字段格式如图6-25所示。

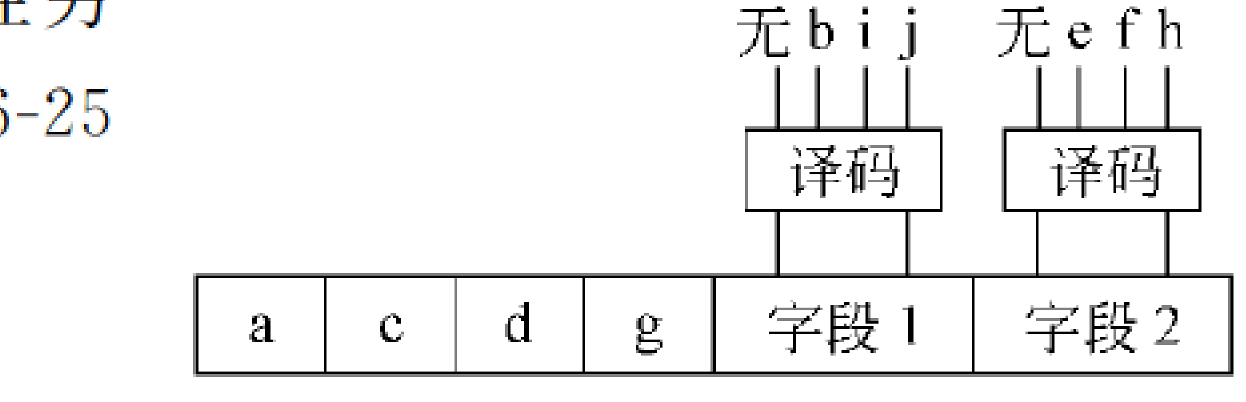


图6-25 微指令控制字段格式

其中,字段1的译码器输出对应的微命令为:

00 无
01 b
10 i
11 j

字段2的译码器输出对应的微命令为:

00 无
01 e
10 f
11 h

将全部8条微指令代码化可以得到:

I_1:11100101
I_2:10110010
I_3:00000111
I_4:01000000
I_5:01011001
I_6:10001111
I_7:01100011
I_8:10000111

【例6.15】 在微程序控制器中,微程序计数器(μPC)可以用具有计数(加1)功能的微地址寄存器(μMAR)来代替,试问程序计数器(PC)是否可以用具有计数功能的存储器地址寄存器(MAR)代替?为什么?

解:在微程序控制器中不可以用MAR来代替PC。因为控存中只有微指令,为了降低成本,可以用具有计数功能的微地址寄存器(μMAR)来代替μPC。而主存中既有指令又有数据,它们都以二进制代码形式出现,取指令和数据时地址的来源是不同的。

取指令:(PC)→MAR
取数据:地址形成部件→MAR

所以,不能用MAR代替PC。

【例6.16】 采用微程序控制器的某计算机在微程序级采用两级流水线,即取第$i+1$条

微指令与执行第 i 条微指令同时进行。假设微指令的执行时间需要 40ns，问：

(1) 控制存储器 CM 选用读出时间为 30ns 的 ROM，问这种情况下微周期为多少？并画出微指令执行时序图。

(2) 若控制存储器 CM 选用读出时间为 50ns 的 ROM，问这种情况下微周期又为多少？并画出微指令执行时序图。

解：(1) 因为取第 $i+1$ 条微指令与执行第 i 条微指令同时进行，取微指令的读出时间为 30ns，而微指令的执行时间需要 40ns，这种情况下微周期取最长的时间，即微周期为 40ns，微指令执行时序图如图 6-26(a)所示。

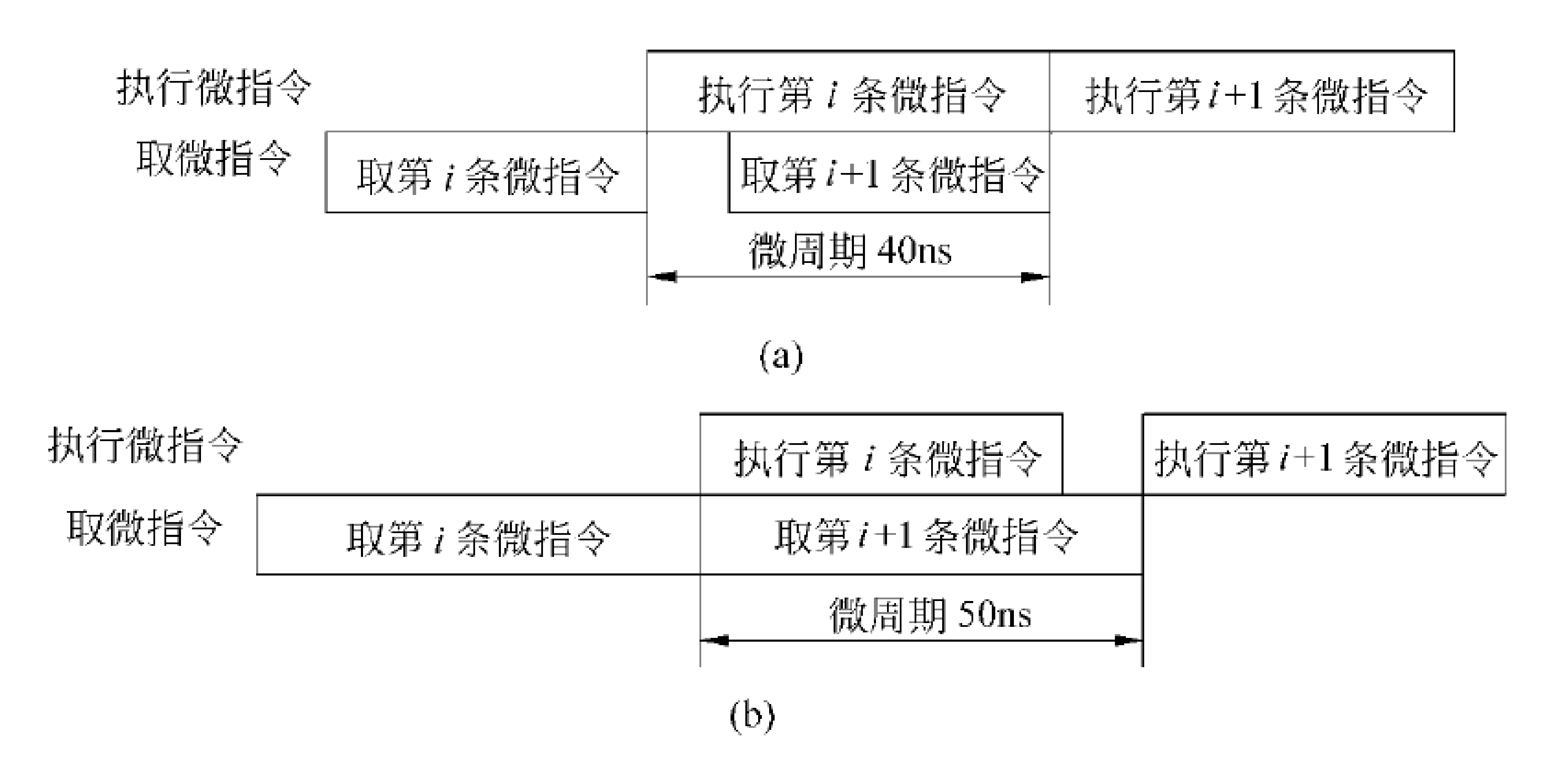


图 6-26 微指令执行时序图

(2) 若控制存储器 CM 选用读出时间为 50ns 的 ROM，这种情况下微周期需取 50ns，微指令执行时序图如图 6-26 (b)所示。

【例 6.17】 某计算机采用 5 级指令流水线，如果每级执行时间是 2ns，求理想情况下该流水线的加速比和吞吐率。

解：流水线的加速比是指采用流水线技术时指令的执行速度与等效的不采用流水线技术的指令执行速度之比，理想情况加速比等于流水线的级数，所以本例中加速比等于 5。

吞吐率指每秒钟能处理的指令数量，现在每完成一条指令的时间是 2ns，则吞吐率等于 $1 \div 2\text{ns} = 0.5 \times 10^9$。

***【例 6.18】** 冯·诺依曼计算机中指令和数据均以二进制形式存放在存储器中，CPU 区分它们的依据是________。

A. 指令操作码的译码结果　　B. 指令和数据的寻址方式

C. 指令周期的不同阶段　　D. 指令和数据所在的存储单元

解：C。

分析：在冯·诺依曼结构计算机中指令和数据均以二进制形式存放在同一个存储器中，CPU 可以根据指令周期的不同阶段来区分是指令还是数据，通常在取指阶段取出的是指令，其他阶段取出的是数据。区分指令和数据还有一个方法，即取指令和取数据时地址的来源是不同的，指令地址来源于程序计数器 PC，而数据地址来源于地址形成部件或指令的地址码字段。

本题较容易误选为 A，需要搞清楚的是，CPU 只有在确定取出的是指令之后，才会将其操作码部分送去译码，因此是不可能依据译码的结果来区分指令和数据的。

*【例 6.19】 下列关于 RISC 的叙述中,错误的是________。

A. RISC 普遍采用微程序控制器

B. RISC 大多数指令在一个时钟周期内完成

C. RISC 的内部通用寄存器数量相对 CISC 多

D. RISC 的指令数、寻址方式和指令格式种类相对 CISC 少

解:A。

分析:B、C、D 3 个选项都是 RISC 的特点之一,所以它们都是正确的,只有 A 选项是错误的,因为 RISC 的速度快,所以普遍采用硬布线控制器,而非微程序控制器。

*【例 6.20】 某计算机的指令流水线由 4 个功能段组成,指令流经各功能段的时间(忽略各功能段之间的缓存时间)分别为 90ns、80ns、70ns 和 60ns,则该计算机的 CPU 时钟周期至少是________。

A. 90ns　　B. 80ns　　C. 70ns　　D. 60ns

解:A。

分析:这个指令流水线的各功能段执行时间是不相同的。由于各功能段的时间不同,计算机的 CPU 时钟周期应当以最长的功能段执行时间为准,也就是说,当流水线充满之后,每隔 90ns 可以从流水线中流出一条指令(假设不存在断流)。

*【例 6.21】 相对于微程序控制器,硬布线控制器的特点是________。

A. 指令执行速度慢,指令功能的修改和扩展容易

B. 指令执行速度慢,指令功能的修改和扩展难

C. 指令执行速度快,指令功能的修改和扩展容易

D. 指令执行速度快,指令功能的修改和扩展难

解:D。

分析:在同样的半导体工艺条件下,硬布线(组合逻辑)控制器的速度比微程序控制器的速度快。这是因为硬布线控制器的速度主要取决于逻辑电路的延迟,而微程序控制器增加了一级控制存储器,执行的每条微指令都要从控存中读取,影响了速度。由于硬布线控制器一旦设计完成就很难改变,所以指令功能的修改和扩展难。

*【例 6.22】 下列寄存器中,汇编语言程序员可见的是________。

A. 存储器地址寄存器(MAR)　　B. 程序计数器(PC)

C. 存储器数据寄存器(MDR)　　D. 指令寄存器(IR)

解:B。

分析:在 CPU 的专用寄存器中,只有 PC 和 PSWR 是汇编语言程序员可见的,而 IR、MAR 和 MBR 对于汇编语言程序员来说是不可见的,它们是 CPU 的内部工作寄存器,对于汇编语言程序员来说是透明的,在汇编语言程序设计中不会出现。

*【例 6.23】 下列选项中,不会引起指令流水线阻塞的是________。

A. 数据旁路(转发)　　B. 数据相关

C. 条件转移　　D. 资源冲突

解:A。

分析:有三种相关可能引起指令流水线阻塞:①结构相关,又称资源相关;②数据相

关；③控制相关，又称为指令相关，主要由转移指令引起。而选项A指出的数据旁路技术，又称为定向技术或相关专用通路技术。其主要思想是不必等待某条指令的执行结果送回到寄存器后，再从寄存器中取出该结果作为下一条指令的源操作数，而是直接将执行结果送到其他指令所需要的地方，这样可以使流水线不发生停顿。

*【例6.24】 下列给出的指令系统特点中，有利于实现指令流水线的是________。

Ⅰ. 指令格式规整且长度一致

Ⅱ. 指令和数据按边界对齐存放

Ⅲ. 只有Load/Store指令才能对操作数进行存储访问

A. 仅Ⅰ、Ⅱ　　B. 仅Ⅱ、Ⅲ　　C. 仅Ⅰ、Ⅲ　　D. Ⅰ、Ⅱ、Ⅲ

解：D。

分析：特点Ⅰ和Ⅲ都是RISC机的特征，而特点Ⅱ则有利于指令和数据的存放，所以以上3个特点都有利于实现指令流水线。

*【例6.25】 假定不采用Cache和指令预取技术，且机器处于“开中断”状态，则在下列有关指令执行的叙述中，错误的是________。

A. 每个指令周期中CPU都至少访问内存一次

B. 每个指令周期一定大于或等于一个CPU时钟周期

C. 空操作指令的指令周期中任何寄存器的内容都不会被改变

D. 当前程序在每条指令执行结束时都可能被外部中断打断

解：C。

分析：本题涉及的概念比较多。首先，如果不采用Cache和指令预取技术，每个指令周期中至少要访问内存一次，即从内存中取指令。其次，指令有的简单有的复杂，每个指令周期总大于或等于一个CPU时钟周期。再次，即使是空操作指令，在指令周期中程序计数器PC的内容也会改变(PC值加“1”)，为取下一条指令做准备。最后，如果机器处于“开中断”状态，在每条指令执行结束时都可能被新的更高级的中断请求所打断。

*【例6.26】 某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用字段直接编码法，共有33个微命令，构成5个互斥类，分别包含7、3、12、5和6个微命令，则操作控制字段至少有________。

A. 5位　　B. 6位　　C. 15位　　D. 33位

解：C。

分析：假设某字段共有N个互斥的微命令，则字段的长度$n \geq \log_2(N+1)$。33个微命令分成5个互斥类(即5个字段)，根据每个类中微命令的多少可以分别确定字段的长度为3、2、4、3、3位，它们之和就是操作控制字段的位数。

*【例6.27】 某CPU主频为1.03GHz，采用4级指令流水线，每个流水线的执行需要1个时钟周期。假定CPU执行了100条指令，在其执行过程中，没有发生任何流水线阻塞，此时流水线的吞吐率为________。

A. 0.25×10^9条指令/秒　　B. 0.97×10^9条指令/秒

C. 1.0×10^9条指令/秒　　D. 1.03×10^9条指令/秒

解:C。

分析:时钟周期为主频的倒数。对于 CPU 主频为 1.03GHz 的 4 级指令流水线,CPU 执行 100 条指令的时间为 $4\times\frac{1}{1.03\times10^9}+99\times\frac{1}{1.03\times10^9}=100\times10^{-9}$,实际吞吐率为 1.0×10^9 条指令/秒。

*【**例 6.28**】 某计算机采用微程序控制器,共有 32 条指令,公用的取指微程序包含 2 条微指令,各指令对应的微程序平均由 4 条微指令组成,采用断定法(下址字段法)确定下条微指令地址,则微指令中下址字段的位数至少是________。

A. 5　　B. 6　　C. 8　　D. 9

解:C。

分析:32 条机器指令,每条指令对应的微程序由 4 条微指令组成,控制存储器需要 128 个单元,再加上公用的取指微程序,所以控存大小至少要有 256 个单元,下址字段的位数至少有 8 位。

*【**例 6.29**】 某计算机字长 16 位,采用 16 位定长指令字结构,部分数据通路结构如图 6-27 所示,图中所有控制信号为 1 时表示有效,为 0 时表示无效,例如控制信号 MDRinE 为 1 表示允许数据从 DB 输入 MDR,MDRin 为 1 表示允许数据从内总线输入 MDR。假设 MAR 的输出一直处于使能状态。加法指令“ADD(R_1),R_0”的功能为 $(R_0)+((R_1))\to(R_1)$,即将 R_0 中的数据与 R_1 的内容所指主存单元的数据相加,并将结果送入 R_1 的内容所指主存单元中保存。

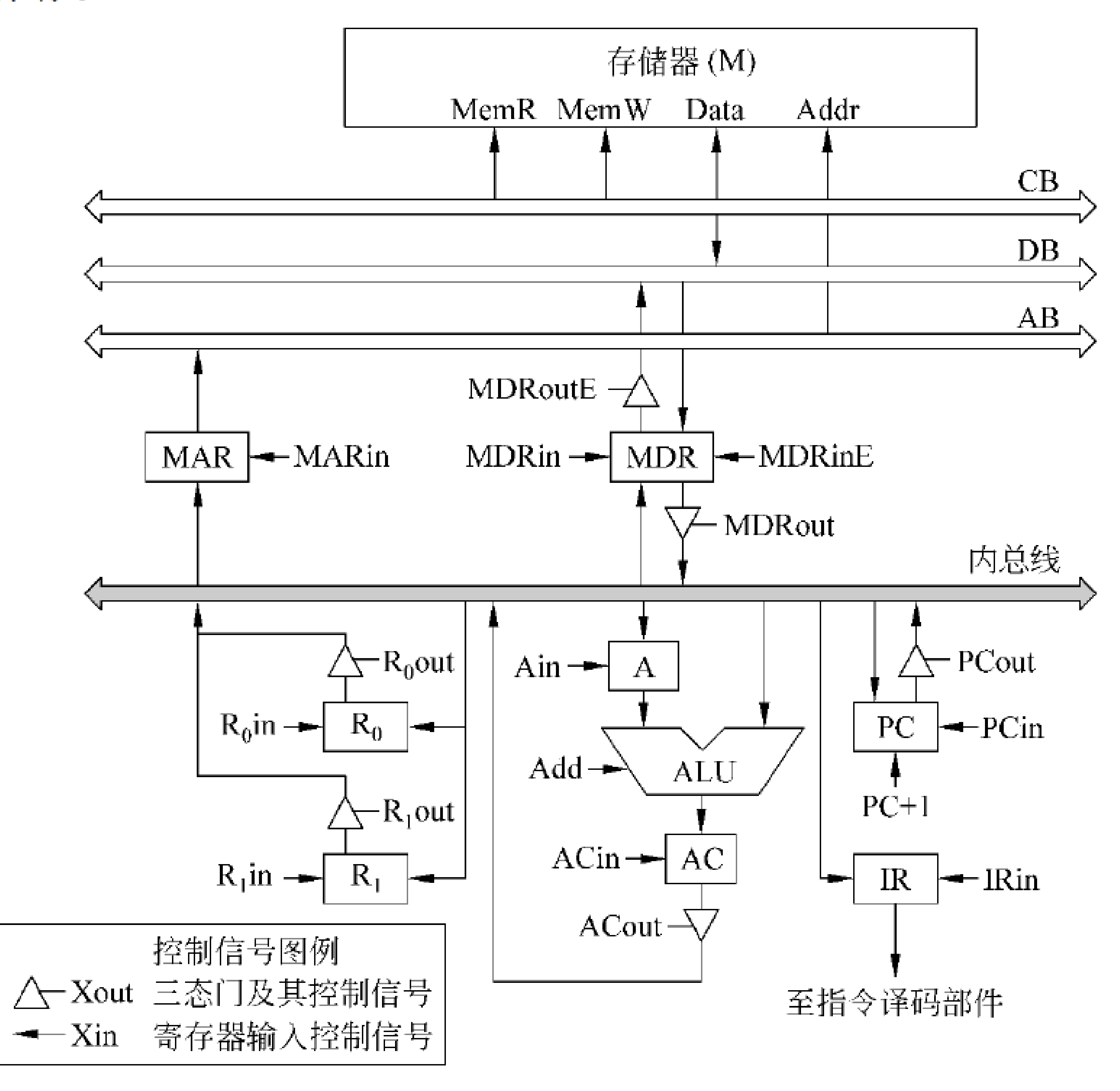


图 6-27　数据通路结构

表 6-6 给出了上述指令取指和译码阶段每个节拍(时钟周期)的功能和有效控制信号，请按表中描述方式用表格列出指令执行阶段每个节拍的功能和有效控制信号。

解： 表 6-7 列出执行阶段每个节拍(时钟周期)的功能和有效控制信号。

表 6-6　取指和译码阶段的功能和有效控制信号

时　钟	功　　能	有效控制信号
C_1	MAR←(PC)	PCout，MARin
C_2	MDR←M(MAR) PC←(PC)+1	MemR，MDRinE PC+1
C_3	IR←(MDR)	MDRout，IRin
C_4	指令译码	无

表 6-7　执行阶段的功能和有效控制信号

时　钟	功　　能	有效控制信号
C_5	MAR←(R_1)	R_1out，MARin
C_6	MDR←M(MAR)	MemR，MDRinE
C_7	A←(MDR)	MDRout，Ain
C_8	AC←(R_0)+(A)	R_0out，Add，ACin
C_9	MDR←(AC)	ACout，MDRin
C_{10}	M←(MDR)	MDRoutE，MemW

分析： 数据通路是 CPU 中算术逻辑单元(ALU)、寄存器(专用和通用)以及存储器之间的连接线路。不同计算机的数据通路可能是不同的，一般在题干中都会给出相应的数据通路结构图。只有明确了机器的数据通路，才能确定相应的微操作控制信号。

在图 6-27 中各部件名称用大写字母表示，各部件名称后加 in 表示该部件的接收控制信号，实际上就是该部件的输入开门信号；各部件名称后加 out 表示该部件的发送控制信号，实际上就是该部件的输出开门信号。由于该机 CPU 内部采用单总线结构，所以此题的关键是要考虑总线冲突的问题，相应的微操作控制信号必须与给出的数据通路结构一致。

由于此题的题干已经给出了取指和译码阶段每个节拍(时钟周期)的功能和有效控制信号，其中译码阶段比较简单，只需将取出指令的操作码字段送到指令译码器中去译码即可，所以搞清楚取指阶段中数据通路的信息流动顺序和方向就成为突破口，只要读懂了取指阶段的功能和有效控制信号，写出执行阶段的功能和有效控制信号就不是一件难事了。

在 C_1 节拍，打开 PC 的发送控制信号和 MAR 的接收控制信号，即完成指令地址送 MAR 的功能，在 C_2 节拍，发读命令，允许数据(此时就是读出的指令)从 DB 输入 MDR，同时 PC 的内容自动加 1，在 C_3 节拍，打开 MDR 的发送控制信号和 IR 的接收控制信号，即完成取出的指令送指令寄存器的功能。

根据加法指令“ADD(R_1)，R_0”的功能(R_0)+((R_1))→(R_1)可知，参加运算的一个操作数在主存中，另一个操作数在寄存器中，结果存放在主存中。C_5～C_7 节拍完成主存中取操作数的功能，其控制信号与取指令阶段的控制信号相似，不同之处在于：①数据地址来自于寄存器 R_1；②取出的数据存放于寄存器 A。C_8 节拍，完成加法运算，运算结果送寄存器 AC。C_9～C_{10} 节拍完成将加法结果写回 R_1 的内容所指主存单元中的功能，由于 MAR 中的内容(R_1 的内容)并没有改变，在 C_9 节拍，只需要打开 AC 的发送控制信号和 MDR 的接收控制信号(将写入的数据送 MDR)。在 C_{10} 节拍，允许数据从 MDR 输入 DB，并发写命令，将数据写入主存单元。

*【例 6.30】 某 16 位计算机中，带符号整数用补码表示，数据 Cache 和指令 Cache 分离。表 6-8 给出了指令系统中部分指令格式，其中 Rs 和 Rd 表示寄存器，mem 表示存储单元地址，(x)表示寄存器 x 或存储单元 x 的内容。

表 6-8　指令系统中部分指令格式

名　　称	指令的汇编格式	指 令 功 能
加法指令	ADD　Rs，Rd	(Rs)+(Rd)→Rd
算术/逻辑左移	SHL　Rd	2 * (Rd)→Rd
算术右移	SHR　Rd	(Rd)/2→Rd
取数指令	LOAD　Rd，mem	(mem)→Rd
存数指令	STORE　Rs，mem	(Rs)→mem

该计算机采用5段流水方式执行指令，各流水段分别是取指(IF)、译码/读寄存器(ID)、执行/计算有效地址(EX)、访问存储器(M)和结果写回寄存器(WB)，流水线采用“按序发射，按序完成”方式，没有采用转发技术处理数据相关，并且同一个寄存器的读和写操作不能在同一个时钟周期内进行。请回答下列问题。

(1) 若int型变量x的值为−513，存放在寄存器R_1中，则执行指令“SHR R_1”后，R_1的内容是多少？(用十六进制表示)

(2) 若某个时间段中，有连续的4条指令进入流水线，在其执行过程中没有发生任何阻塞，则执行这4条指令所需的时钟周期数为多少？

(3) 若高级语言程序中某赋值语句为$x=a+b$，x、a和b均为int型变量，它们的存储单元地址分别表示为$[x]$、$[a]$和$[b]$。该语句对应的指令序列及其在指令流水线中的执行过程如图6-28所示。

I_1　LOAD　R_1，[a]
I_2　LOAD　R_2，[b]
I_3　ADD　R_1，R_2
I_4　STORE　R_2，[x]

	时间单元													
指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I_1	IF	ID	EX	M	WB									
I_2		IF	ID	EX	M	WB								
I_3			IF				ID	EX	M	WB				
I_4							IF				ID	EX	M	WB

图 6-28　指令序列及其执行过程示意图

则这4条指令执行过程中，I_3的ID段和I_4的IF段被阻塞的原因各是什么？

(4) 若高级语言程序中某赋值语句为$x=2*x+a$，x和a均为unsigned int类型变量，它们的存储单元地址分别表示为$[x]$、$[a]$，则执行这条语句至少需要多少个时钟周期？要求模仿图6-28画出这条语句对应的指令序列及其在流水线中的执行过程示意图。

解：(1) x的值为−513，对应二进制为−1000000001，存放在16位的寄存器R_1中。指令执行前，(R_1)=1111 1101 1111 1111B=FDFFH。执行指令“SHR R_1”即右移1位后，

(R_1) = 1111 1110 1111 1111B = FEFFH。

(2) 设一个 m 段流水线的各段经过时间均为 Δt_0，则需要 $T_0 = m\Delta t_0$ 的流水建立时间，之后每隔 Δt_0 就可流出一条指令，完成 n 个任务的解释共需时间 $T = m\Delta t_0 + (n-1)\Delta t_0$。连续 4 条指令流入流水线，且不考虑阻塞问题，至少需要 5+(4−1)=8 个时钟周期数。

(3) 完成 $x=a+b$ 功能的 4 条指令中会出现数据相关的问题，某些段会出现阻塞。I_3 的 ID 段被阻塞的原因：因为 I_3 与 I_1 和 I_2 都存在数据相关，需等到 I_1 和 I_2 将结果写回寄存器后，I_3 才能读寄存器内容，所以 I_3 的 ID 段被阻塞。I_4 的 IF 段被阻塞的原因：因为 I_4 的前一条指令 I_3 在 ID 段被阻塞，所以 I_4 的 IF 段被阻塞。

(4) 模仿图 6-28 画出 $x=2*x+a$ 对应的指令序列及其在流水线中的执行过程示意图。$2*x$ 操作可以由 x 左移 1 位或 x 加 x 两种方法实现。

```
x=2*x+a对应的指令序列为
I1   LOAD   R1,[x]
I2   LOAD   R2,[a]
I3   SHL    R1              //或者 ADD   R1,R1
I4   ADD    R1,R2
I5   STORE  R2,[x]
```

这 5 条指令在流水线中的执行过程如图 6-29 所示，执行 $x=2*x+a$ 语句最少需要 17 个时钟周期。

	时 间 单 元																
指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I_1	IF	ID	EX	M	WB												
I_2		IF	ID	EX	M	WB											
I_3			IF			ID	EX	M	WB								
I_4						IF				ID	EX	M	WB				
I_5										IF				ID	EX	M	WB

图 6-29　5 条指令在流水线中的执行过程示意图

第(4)问的答案并不唯一，只要能实现 $x=2*x+a$ 的功能即可。例如，如果上述 5 条指令中的 I_2 和 I_3 对调，同样能实现 $x=2*x+a$ 的功能，但由于数据相关的原因，最少需要 18 个时钟周期。这里有个指令序列优化的问题，可以使执行时间最少。

分析：本题是一道涉及多个知识点的综合题。需要对汇编语言指令和指令流水线的概念都很清晰。

6.4　同步测试习题及解答

6.4.1　同步测试习题

一、填空题

1. 控制器由于设计方法的不同可分为________型、________型和________型控制器。

2. 控制器在生成各种控制信号时，必须按照一定的________进行，以便对各种操作实施时间上的控制。

3. 微程序控制的计算机中的控制存储器CM是用来存放________的。

4. 在微指令的字段编码法中，操作控制字段的分段并非是任意的，必须遵循的分段原则中包括：①把________性的微命令分在同一段内；②一般每个小段要留出一个状态，表示________。

5. 微指令分为________和________两类，________微指令可以同时执行若干个微操作，所以执行机器指令的速度比________微指令快。

二、选择题

1. 在CPU中跟踪指令后继地址的寄存器是________。
   A. 主存地址寄存器　　B. 程序计数器
   C. 指令寄存器　　D. 状态标志寄存器

2. 指令寄存器的位数取决于________。
   A. 存储器的容量　　B. 指令字长
   C. 机器字长　　D. 存储字长

3. 在计算机系统中，表征系统运行状态的部件是________。
   A. 程序计数器　　B. 累加寄存器
   C. 中断寄存器　　D. 程序状态字

4. 通用寄存器是________。
   A. 可存放指令的寄存器
   B. 可存放程序状态字的寄存器
   C. 本身具有计数逻辑与移位逻辑的寄存器
   D. 可编程指定多种功能的寄存器

5. 指令译码器是对________进行译码。
   A. 整条指令　　B. 指令的操作码字段
   C. 指令的地址　　D. 指令的操作数字段

6. 微操作信号发生器的作用是________。
   A. 从主存中取出指令　　B. 完成指令操作码的分析功能
   C. 产生控制时序　　D. 产生各种微操作控制信号

7. 下列说法中________是正确的。
   A. 指令周期等于机器周期　　B. 指令周期小于机器周期
   C. 指令周期大于机器周期　　D. 指令周期是机器周期的两倍

8. 三级时序系统提供的三级时序信号是________。
   A. 指令周期、机器周期、节拍　　B. 指令周期、机器周期、时钟周期
   C. 机器周期、节拍、脉冲　　D. 指令周期、微指令周期、时钟周期

9. 采用同步控制的目的是________。
   A. 提高执行速度　　B. 简化控制时序
   C. 满足不同操作对时间安排的需要　　D. 满足不同设备对时间安排的需要

10. 异步控制常用于________。

A. CPU 访问外围设备时　　B. 微程序控制器中
C. CPU 的内部控制中　　D. 主存的内部控制中

11. 下列叙述正确的是________。
A. 同一 CPU 周期中，可以并行执行的操作称为兼容性微操作
B. 同一 CPU 周期中，不可以并行执行的操作称为兼容性微操作
C. 同一 CPU 周期中，可以并行执行的操作称为互斥性微操作
D. 同一 CPU 周期中，不可以并行执行的操作称为互斥性微操作

12. 下列说法正确的是________。
A. 采用微程序控制器是为了提高速度
B. 控制存储器采用高速 RAM 电路组成
C. 微指令计数器决定指令执行顺序
D. 一条微指令放在控制存储器的一个单元中

13. 下列说法中正确的是________。
A. 微程序控制方式与硬布线控制方式相比较，前者可以使指令的执行速度更快
B. 若采用微程序控制方式，则可用 μPC 取代 PC
C. 控制存储器可以用掩膜 ROM、EPROM 或闪速存储器实现
D. 指令周期也称为 CPU 周期

14. 下列叙述中正确的是________。
A. 控制器产生的所有控制信号称为微指令
B. 微程序控制器比硬布线控制器更加灵活
C. 微处理器的程序称为微程序
D. 采用微程序控制器的处理器称为微处理器

15. 微程序控制器的速度比硬布线控制器慢，主要是因为________。
A. 增加了从磁盘存储器读取微指令的时间
B. 增加了从主存储器读取微指令的时间
C. 增加了从指令寄存器读取微指令的时间
D. 增加了从控制存储器读取微指令的时间

16. 硬布线控制器与微程序控制器相比________。
A. 硬布线控制器的时序系统比较简单
B. 微程序控制器的时序系统比较简单
C. 两者的时序系统复杂程度相同
D. 可能是硬布线控制器的时序系统简单，也可能是微程序控制器的时序系统简单

17. 微程序控制器中，控制部件向执行部件发出的某个控制信号称为________。
A. 微程序　　B. 微指令　　C. 微操作　　D. 微命令

18. 微程序控制器中，机器指令与微指令的关系是________。
A. 每一条机器指令由一条微指令来执行
B. 一条机器指令由一段用微指令编成的微程序来解释执行
C. 一段机器指令组成的程序可由一个微程序来执行
D. 每一条微指令由一条机器指令来解释执行

19. 微程序控制器中,微程序的入口地址是由________形成的。

A. 机器指令的地址码字段　　B. 微指令的微地址码字段

C. 机器指令的操作码字段　　D. 微指令的微操作码字段

20. 微指令执行的顺序控制问题,实际上是如何确定下一条微指令的地址问题。通常采用的一种方法是断定方式。其基本思想是________。

A. 用程序计数器PC来产生后继微指令地址

B. 用微程序计数器μPC来产生后继微指令地址

C. 通过微指令顺序控制字段由设计者指定或者由判断字段控制产生后继微指令地址

D. 通过指令中指定一个专门字段来控制产生后继微指令地址

21. 兼容性微命令指几个微命令是________。

A. 可以同时出现的　　B. 可以相继出现的

C. 可以相互替代的　　D. 可以相互容错的

22. 下列不符合RISC特点的是________。

A. 指令长度固定,指令种类少

B. 寻址方式种类丰富,指令功能尽量增强

C. 设置大量通用寄存器,访问存储器指令简单

D. 选取使用频率较高的一些简单指令。

23. 以下关于CISC/RISC计算机的叙述中,错误的是________。

A. RISC机器指令比CISC机器指令简单

B. RISC中通用寄存器比CISC多

C. CISC机器采用微码比RISC多

D. CISC比RISC机器可以更好地支持高级语言

24. 设指令由取指、分析、执行3个子部件完成,并且每个子部件的时间均为Δt,若采用常规标量单流水线处理机(即处理机的度为1),连续执行12条指令,共需________。

A. $12\Delta t$　　B. $14\Delta t$　　C. $16\Delta t$　　D. $18\Delta t$

25. 现有四级指令流水线,分别完成取指、取数、运算、传送结果4步操作。若完成上述操作的时间依次为9ns、10ns、6ns、8ns。则流水线的操作周期应设计为________。

A. 6ns　　B. 8ns　　C. 9ns　　D. 10ns

三、判断题

1. 在冯·诺依曼计算机中,指令流是由数据流驱动的。(　　)
2. 执行指令时,指令在主存中的地址存放在指令寄存器中。(　　)
3. 指令周期是指CPU从主存中读出一条指令的时间。(　　)
4. 指令周期又称为CPU周期。(　　)
5. 取指周期的操作与指令的操作码无关。(　　)
6. 微指令是指控制存储器中的一个单元的内容。(　　)
7. 在微程序控制器中,微指令寄存器用来存放微程序。(　　)
8. 微指令的操作控制字段采用字段编码时,兼容的微命令应安排在同一段中。(　　)

四、简答题

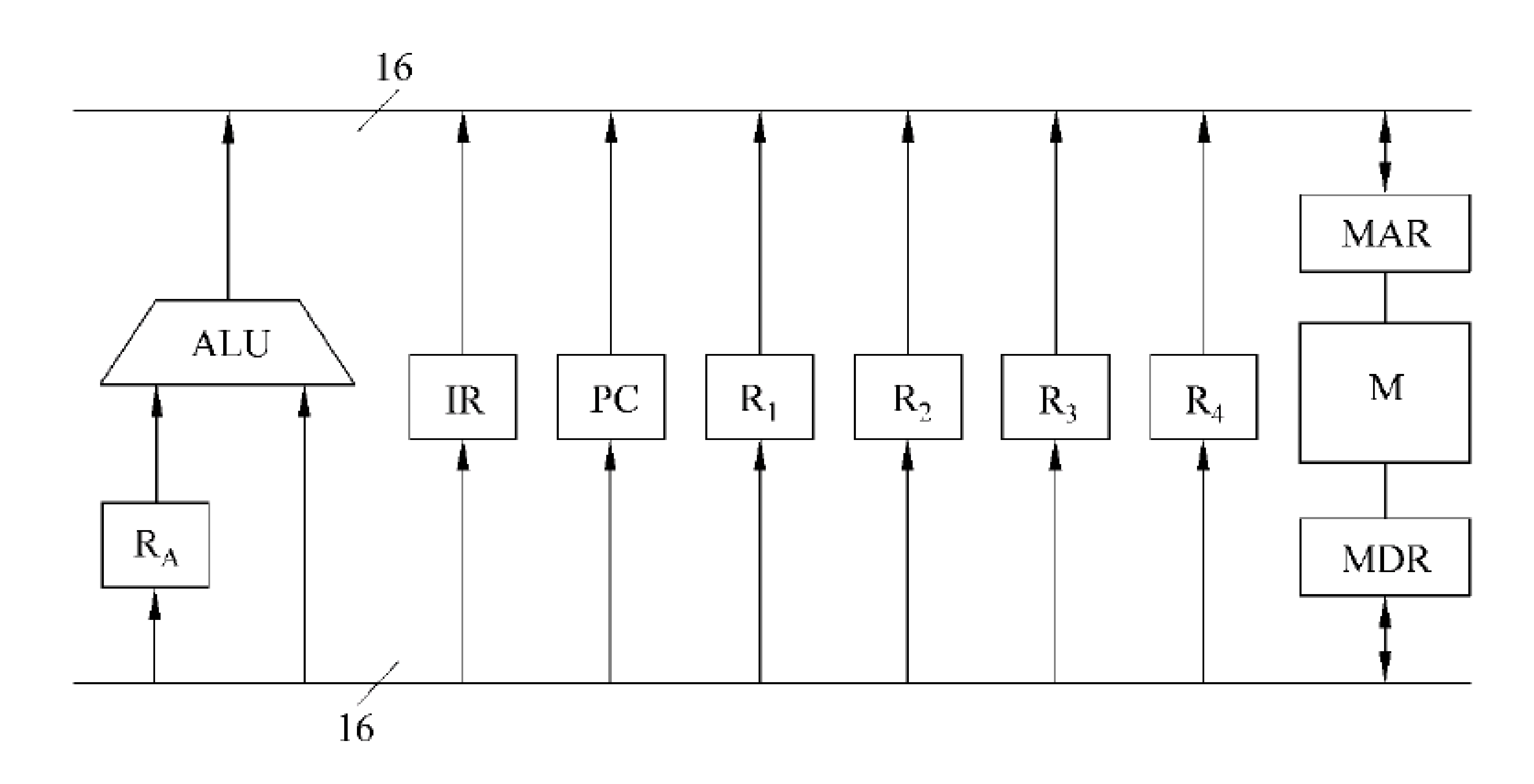
1. 在控制器中，微操作控制信号的形成与哪些信号有关？

2. 微程序控制和组合逻辑控制哪一种速度更快？为什么？

3. 什么是指令周期、机器周期（CPU 周期）和 T 周期？

五、设计题

1. 一个 CPU 数据通路为双总线结构，如图 6-30 所示。其中，图中连线有误。



注：ALU——运算器；RA——ALU 的输入寄存器；IR——指令寄存器；PC——程序计数器；R_1～R_4：程序员可用通用寄存器；MAR——存储器地址寄存器；MDR——存储器数据寄存器。

图 6-30　数据通路示意图

回答下列问题：

（1）画出修正错误后的连线图，不能改变原有的双总线结构。

（2）如要实现直接寻址方式，如何修改？

（3）描述 ADD addr，R_1 指令从取指令开始的实现过程。指令的功能为（R_1）＋（addr）→addr。

2. 某机采用微程序控制方式，微指令字长 24 位，采用水平型编码控制的微指令格式，断定方式。共有微命令 30 个，构成 4 个互斥类，各包含 5 个、8 个、14 个和 3 个微命令，外部条件共 3 个。

（1）控制存储器的容量应为多少？

（2）设计出微指令的具体格式。

3. 一个假想机的数据通路如图 6-31 所示，它的控制存储器容量为 256 个单元。ALU 可完成算术加、减和逻辑与、或运算，ALU 有标志位 Z 和 N，微指令要完成有条件和无条件转移功能。

设计微指令格式，使之能完成上述要求的功能，表明微指令中每一位的符号及其功能。如微指令为多个子周期，有几个子周期？每个子周期完成什么操作？（提示：可考虑寄存器运算微指令和访问主存微指令两种类型微指令，并假定在一个微指令周期内就可以完成 MDR 与主存间的数据传送）

4. 一个 CPU 数据通路为双总线结构，如图 6-32 所示。IR 为指令寄存器；PC 为程序计数器（具有自增 1 功能），M 为主存（受 R/W 信号控制），MAR 为主存地址寄存器，MDR 为

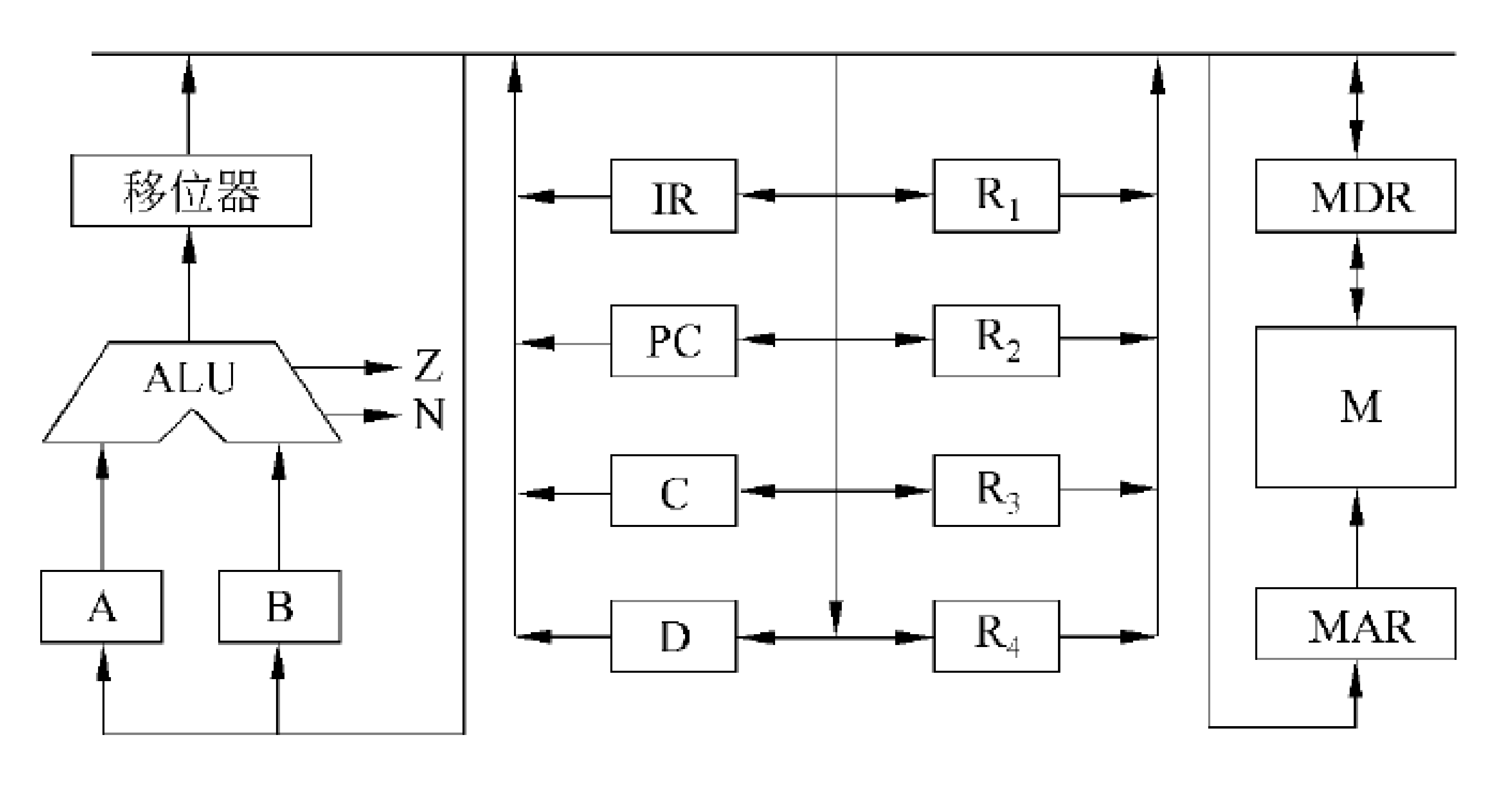


图 6-31　假想机的数据通路

主存数据寄存器，ALU 由＋、－控制信号决定可完成何种操作，G 控制一个门电路，除 MAR、X、Y 的输出端为直通线不受控之外，其余寄存器均有 in、out 控制信号。

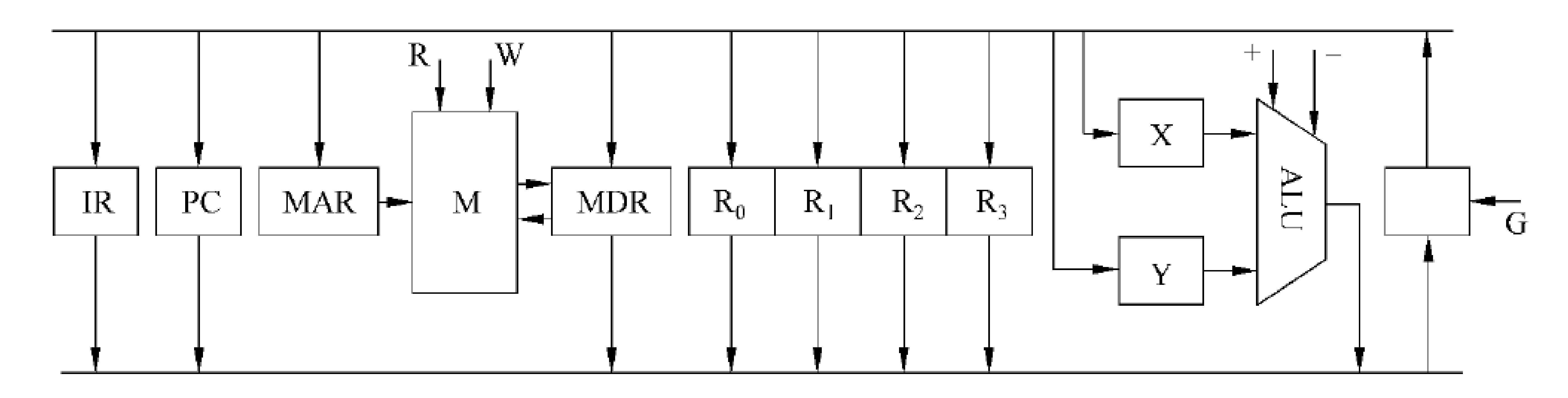


图 6-32　未标出控制信号的双总线结构图

(1) 标出各寄存器的 in、out 控制信号。

(2) 设计微指令格式，并说明各字段意义。

(3) SUB R_2，R_0 指令完成$(R_0)-(R_2)\rightarrow R_0$ 的功能操作，写出该指令从取指令开始的执行过程。

(4) 写出取指令的各条微指令的代码。

5. 某机有 8 条微指令 $I_1 \sim I_8$，每条微指令所含的微命令控制信号如表 6-9 所示。

表 6-9　微指令所含微命令控制信号

微　指　令	微　命　令　信　号									
	a	b	c	d	e	f	g	h	i	j
I_1	√			√						
I_2			√				√		√	
I_3		√				√		√		
I_4	√									√
I_5			√		√				√	
I_6	√			√						√
I_7	√		√							
I_8		√				√		√		

试为 a、b、c、d、e、f、g、h、i、j 这 10 个微命令设计格式并安排编码。

6.4.2 同步测试习题解答

一、填空题

1. 组合逻辑,存储逻辑,组合逻辑和存储逻辑结合。
2. 时序。
3. 微程序。
4. 互斥,本字段不发出任何微命令。
5. 水平型,垂直型,水平型,垂直型。

二、选择题

1. B。程序计数器中存放着正在执行的指令地址或接着要执行的下一条指令地址。

2. B。指令寄存器中存放着正在执行的指令,其位数与指令字长相关。

3. D。程序状态字的各位表征程序和机器运行的状态。

4. D。存放指令的寄存器是指令寄存器(IR),存放程序状态字的寄存器是程序状态字寄存器(PSW),通用寄存器不一定本身具体计数和移位功能。

5. B。指令包括操作码字段和地址码字段,指令译码器(ID)仅对操作码字段进行译码。

6. 微操作信号发生器(CU)用来产生各种微操作控制信号,这些信号是由指令部件提供的译码信号、时序部件提供的时序信号和被控制功能部件所反馈的状态及条件综合形成的。

7. C。一个指令周期可划分为若干个机器周期。

8. C。三级时序系统是指机器周期、节拍、脉冲三级时序信号。

9. B。同步控制采用统一的时钟信号,以最复杂指令的操作时间作为统一的时间间隔标准。这种控制方式设计简单,容易实现。

10. A。异步控制的各项操作不采用统一的时序信号控制,常用于对外设的控制。

11. A。兼容性微操作是指在同一 CPU 周期中,可以并行执行的操作,而互斥性微操作是指不允许同时出现的操作,不可以并行执行的操作并不一定是不允许同时出现的操作。

12. D。微程序控制器比硬布线控制器的速度慢;通常控制存储器采用 ROM 组成;微指令计数器决定的是微指令的执行顺序,只有选项 D 是正确的。

13. C。微程序控制器比硬布线控制器的速度慢;μPC 是微程序计数器,不能取代 PC;CPU 周期又称机器周期,而不是指令周期,所以答案为 C。

14. B。控制器产生的所有控制信号称为微命令;微处理器中的程序还称为程序,不称为微程序;微处理器的控制器可以是微程序控制器也可以是硬布线控制器,所以答案为 B。

15. D。由于微程序控制器增加了控制存储器,故指令的执行速度比硬布线控制器慢。

16. B。微程序控制器的时序系统相对简单。

17. D。在微程序控制器中,控制部件向执行部件发出的控制信号称为微命令。

18. B。一条机器指令的功能通常用许多条微指令组成的序列来实现,这个微指令序列称为微程序。

19. C。当执行完公用的取指微程序从主存中取出机器指令之后,由机器指令的操作码字段指出各个微程序的入口地址(初始微地址)。

20. C。形成后继微地址有增量方式和断定方式两种，其中增量方式由 μPC 形成，断定方式由设计者指定或者由设计者指定的判断字段控制形成。

21. A。兼容性微命令是指那些可以同时产生，共同完成某一些微操作的微命令。

22. B。B 选项是 CISC 的特点。

23. D。从表 6-1 可以看出，前 3 项都是正确的，根据排除法，选择 D。

24. B。单流水线处理机执行 12 条指令的时间为 $[3+(12-1)]\Delta t=14\Delta t$。

25. D。如果流水线每步操作时间不一样，应选最慢的一步的操作时间作为操作周期。

三、判断题

1. ×。在冯·诺依曼计算机中，数据流是由指令流来驱动的。

2. ×。在执行指令时，存放在指令寄存器中的是指令而不是指令的地址。

3. ×。指令周期是指 CPU 从主存中读出指令、分析取数并执行完该指令的全部时间。

4. ×。指令周期是由若干个 CPU 周期组成的。

5. √。

6. √。

7. ×。在微程序控制器中，微指令寄存器用来存放取出的一条微指令。

8. ×。微指令的操作控制字段采用字段编码时，应将互斥的微命令安排在同一段内，兼容的微命令安排在不同的段内。

四、简答题

1. 微操作控制信号是由指令部件提供的译码信号、时序部件提供的时序信号和被控制功能部件所反馈的状态及条件信号综合形成的。

2. 组合逻辑控制速度更快。因为微程序控制器使每条机器指令都转化成为一段微程序并存入一个专门的存储器(控制存储器)中，微操作控制信号由微指令产生，增加了一级控制存储器，所以速度慢。

3. 指令周期是指取指令、分析取数到执行指令所需的全部时间。一个指令周期划分为若干个机器周期(CPU 周期)，每个机器周期完成一个基本操作。一个机器周期中又含有若干个时钟周期(T 周期)，每个 T 周期完成一个微操作。

五、设计题

1. (1) 修正错误后的连线图如图 6-33 所示。

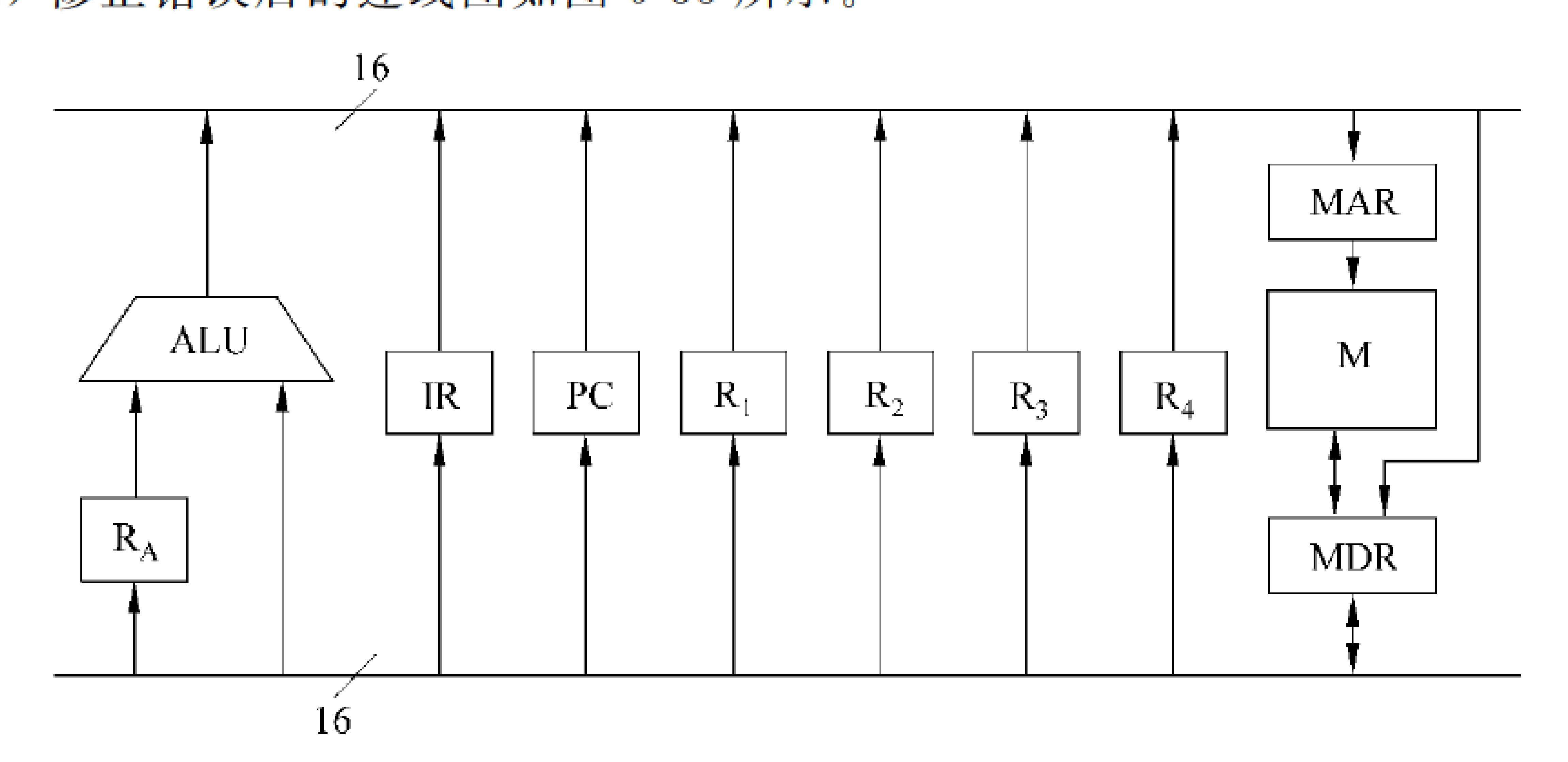


图 6-33　修改后的数据通路示意图

(2) 直接寻址方式就是指令的地址码部分直接给出主存地址,即 IR_{addr}→MAR,原图已有此通路,无须修改。

(3) 指令 ADD addr,R_1 的实现过程。

```
PC→MAR              ;从存储器中取指令
M(MAR)→MDR
MDR→IR
PC+1→PC
IRaddr→MAR          ;从存储器中取加数
M(MAR)→MDR
MDR→RA
R1→MDR              ;从寄存器 R1 中取被加数
+,ALU→MDR           ;求和
MDR→M               ;和写回存储器
```

2. (1) 控制存储器的容量为 256×24。因为下地址字段有 8 位。

(2) 微指令的具体格式见图 6-34。

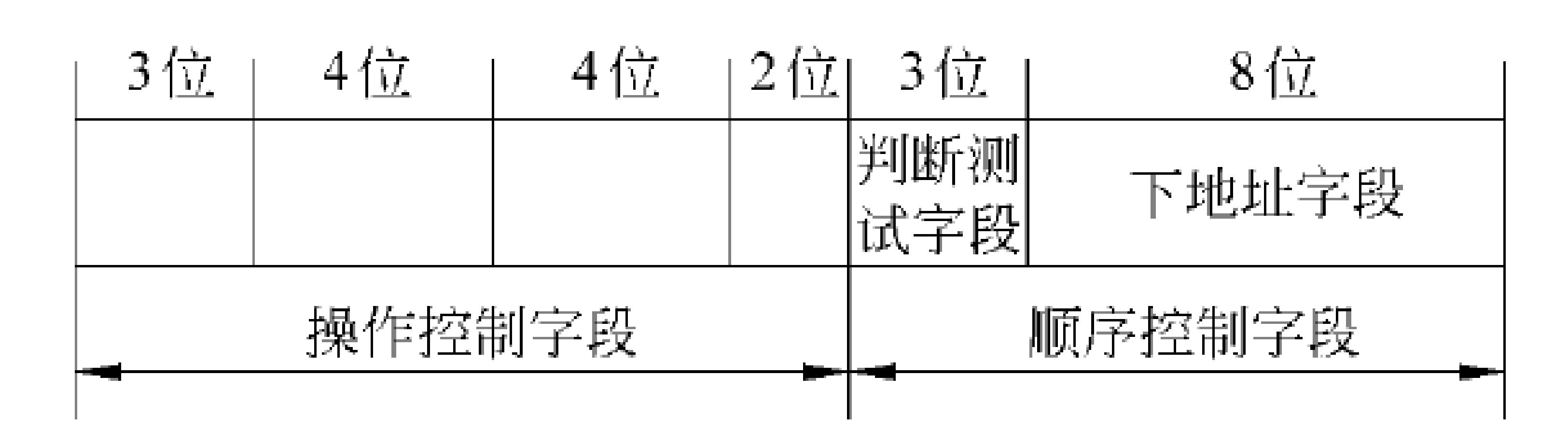


图 6-34　微指令的具体格式

图 6-34 中操作控制字段被分为 4 组,第一组 3 位(表示 5 个微命令),第二组 4 位(表示 8 个微命令),第一组 4 位(表示 14 个微命令),第四组 2 位(表示 3 个微命令);判断测试条件字段 3 位(假设外部条件直接控制),下地址字段 8 位。

3. 微命令包括:

ALU 的控制 4 个(+、−、与、或);

R_0~R_3 的 in、out 信号 8 个;

IR、PC、C、D 的 in、out 信号 8 个;

主存的读写信号 2 个;

MDR 的 in、out 信号 2 个;

MAR、A、B 的 in 信号 3 个。

若微指令采用直接控制法,操作控制字段就需要 27 位。另有判断测试字段 2 位,下地址字段 8 位。微指令格式图略。

寄存器运算微指令有两个子周期:取微指令子周期、执行子周期;访问主存微指令有 3 个子周期:取微指令子周期、访问主存子周期、执行子周期。

4. (1) 假设输入用字母 in 表示,输出用字母 o 表示,标出各寄存器的 in、out 控制信号后的双总线结构如图 6-35 所示。

(2) 若微指令格式中操作控制字段采用直接控制法,则所有微命令每个一位,微指令格式图略。

(3) SUB R_2,R_0 指令的执行过程:

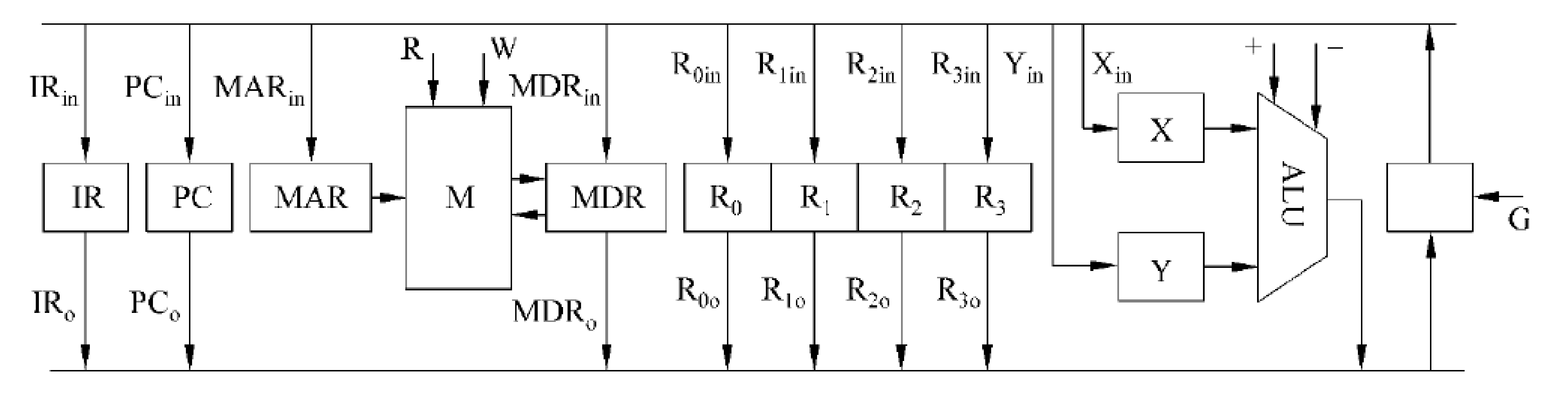


图 6-35　标出控制信号的双总线结构图

PC→MAR

M(MAR)→MDR

MDR→IR

R_0→X

R_2→Y

X-Y→R_0

(4) 写出取指令的各条微指令的代码。

PC→MAR(PC_o,G,MAR_{in})

M(MAR)→MDR(R)

MDR→IR(MDR_o,G,IR_{in})

括号中为各条微指令对应的微命令,具体代码省略。

5. 从表 6-8 可以得出:

b、c、d,e、f、g,h、i、j 分别两两互斥,所以微指令格式如图 6-36 所示。

2位	2位	2位	1位
00 不操作	00 不操作	00 不操作	0 不操作
01 b	01 e	01 h	1 a
10 c	10 f	10 i	
11 d	11 g	11 j	

图 6-36　微指令操作控制字段的格式

I_1～I_8 这 8 条微指令的编码为:

I_1　11 00 00 1

I_2　10 11 10 0

I_3　01 10 01 0

I_4　00 00 11 1

I_5　10 01 10 0

I_6　11 00 11 1

I_7　10 00 00 1

I_8　01 10 01 0

第7章 总线

7.1 基本内容摘要

- 总线概述
  - 总线的基本概念

    三态门；

    总线事务；

    总线使用权。
  - 总线的分类
  - 总线的组成及性能指标
- 总线仲裁
  - 集中仲裁方式

    链式查询；

    计数器定时查询；

    独立请求。
  - 分布仲裁方式
- 总线定时控制
  - 同步定时方式
  - 异步定时方式
- 总线标准
  - 系统总线标准
  - 外部总线标准

7.2 重点难点梳理

1. 三态门

三态门是具有3种逻辑状态的门电路。这3种状态为逻辑“0”、逻辑“1”和浮空状态。所谓浮空状态，就是三态门的输出呈现开路的高阻状态。三态门除了正常的输入端和输出端之外，还有一个控制端G(或$\overline{G}$)。只有当控制端有效时，该三态门才能满足正常的逻辑关系；否则，输出将呈现高阻状态，相当于这个三态门与外界断开联系。

三态门主要用于总线连接,各个部件或设备必须通过三态缓冲器才能挂在总线上,通过控制端选择工作部件或设备。

2. 总线事务类型

通常把在总线上一对设备之间的一次信息交换过程称为一个“总线事务”。总线事务类型通常根据它的操作性质来定义,典型的总线事务类型有“存储器读”、“存储器写”、“I/O读”、“I/O写”、“中断响应”等,一次总线事务简单地说包括地址阶段和数据阶段两个阶段。

有些总线事务要求完成一连串连续单元的读写,如从存储器读出一个Cache行或者写一个Cache行到主存。在这种情况下,一个总线事务能完成多个数据的读写,称为突发传输方式。突发传送事务由一个地址阶段和多个数据阶段构成,用于传送多个连续单元的数据,地址阶段送出的是连续区域的首地址。

3. 总线结构

单总线结构中,所有主要功能部件(如CPU、主存和各I/O接口模块)都挂接在一个总线上。这种总线结构简单,便于扩充,但所有传送都共享一组总线,使总线成为整个系统的瓶颈。因为一个总线上某一时刻只能有一对设备进行传输,故所有设备只能分时共享总线。随着计算机应用领域的扩大,挂接在系统中的外设种类和数量越来也多,对数据传输的速度要求也越来越高,如果还用单总线结构,性能会急剧下降。

在单总线的基础上再开辟一条CPU与主存之间的通路,形成以主存储器为中心的双总线结构,如图7-1(a)所示,存储总线只在主存和CPU之间传输信息,速度快,效率高。另一种双总线结构是一种分层的总线结构,采用输入输出处理器(IOP)方式进行I/O传送,如图7-1(b)所示,其基本思想是将I/O设备从单总线分离出来,减轻了CPU参与I/O的负担,CPU、主存和IOP之间的信息传送是在主存总线上进行的,而各种I/O设备与主机之间的信息交换则通过I/O总线和主存总线进行。

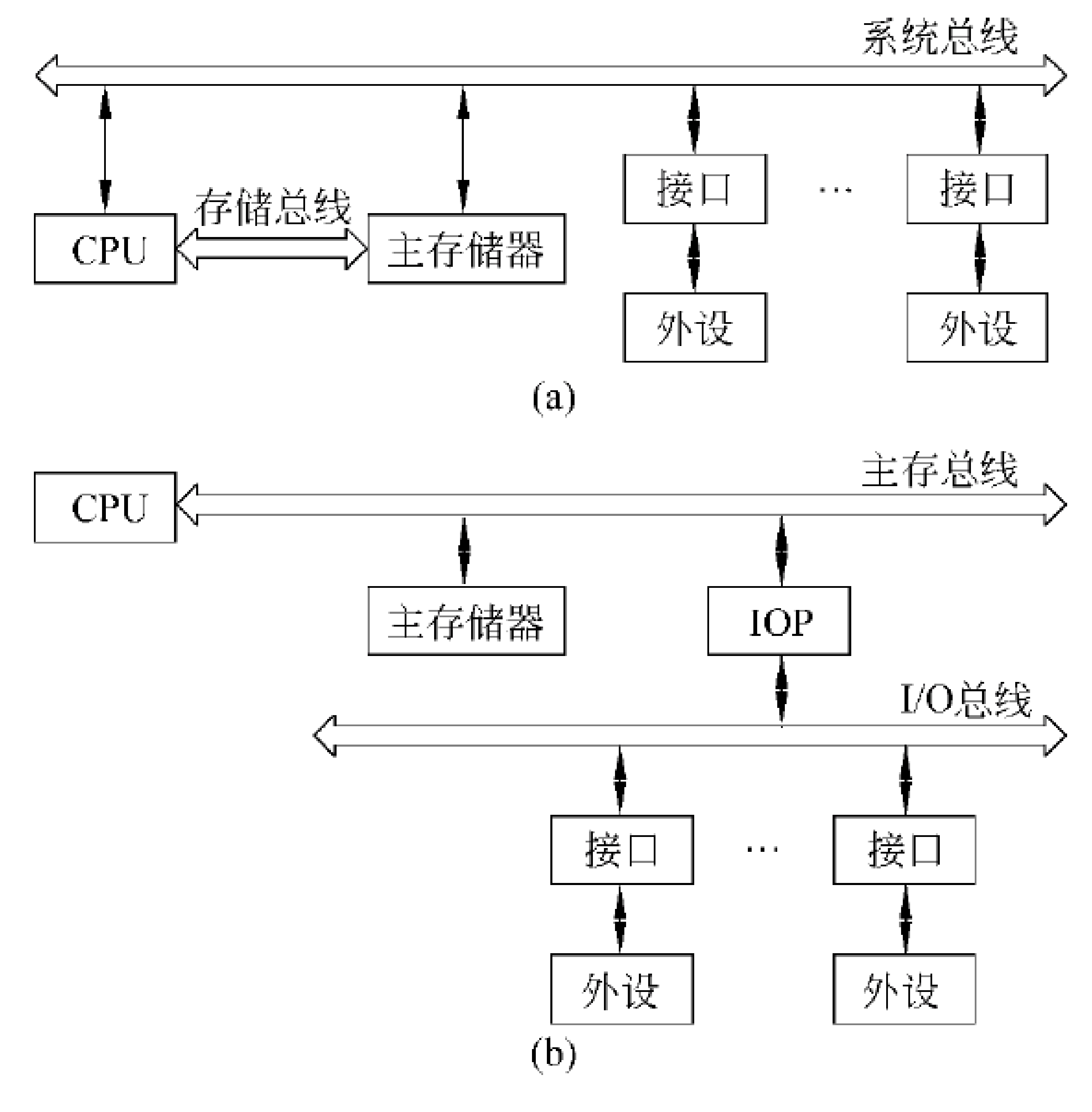


图7-1　两种双总线结构

早期的三总线结构实际是前述两种双总线结构结合的产物,随着计算机系统中相互相

接的部件和设备种类的增加,用于连接部件和设备的系统总线的数量也越来越多。由于大量高性能外设的不断涌现,如果将高速设备与低速设备连在同一个总线上,势必会影响系统的效率,故目前的多级总线结构都将高速I/O设备总线和低速I/O设备总线分离开。图7-2是一个典型的现代微机系统的总线结构,它反映了处理器总线、存储器总线、PCI总线以及外设接口连接CPU、主存和各种外设的连接关系。从图7-2可以看出,越靠近CPU的总线越快,越远离CPU的总线越慢。

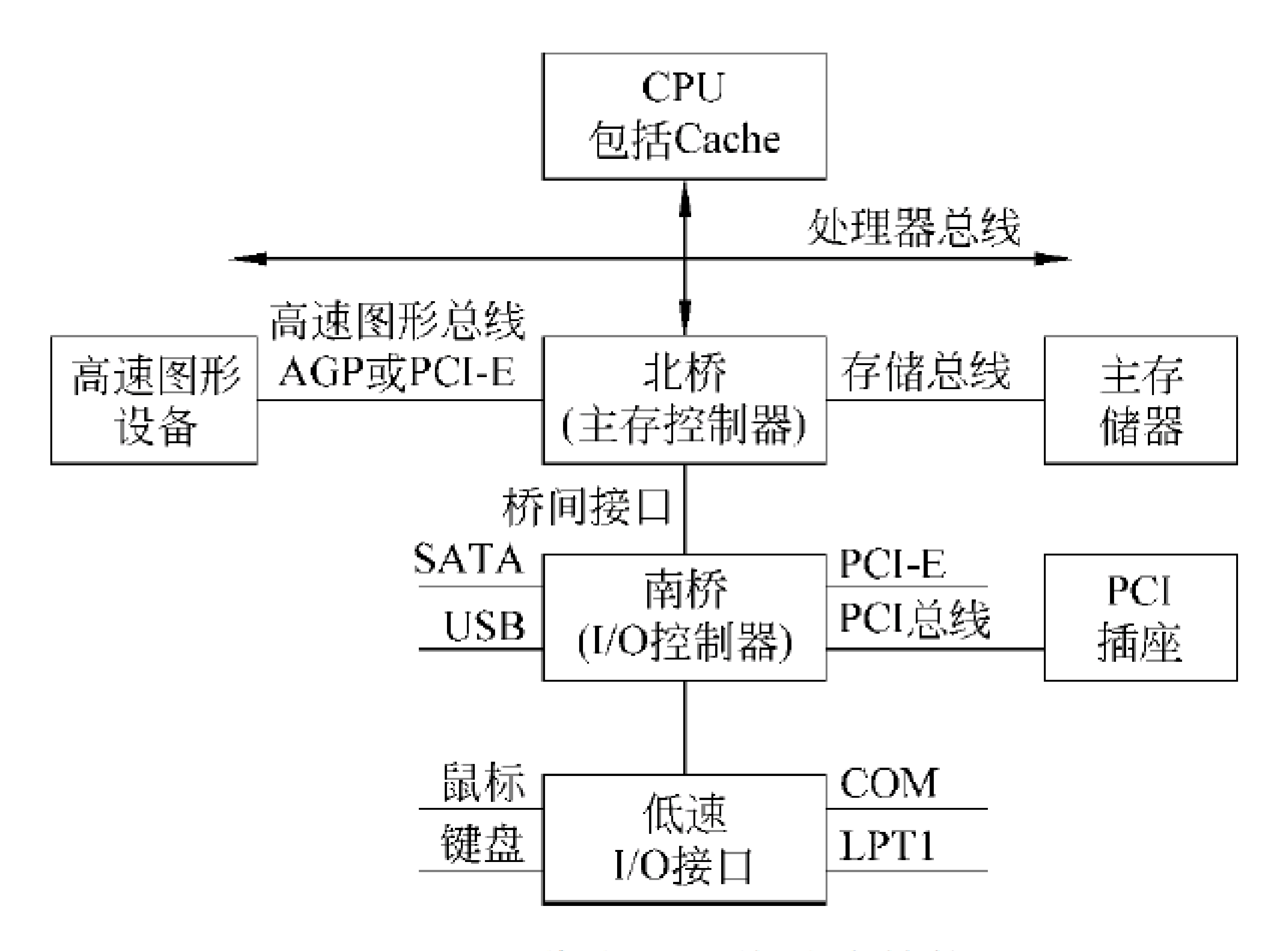


图7-2　现代微机系统总线结构

4. 集中式控制的总线管理

集中式总线控制方式有以下几种。

(1) 链式查询方式

链式查询方式的总线控制器使用3根控制线与所有部件和设备相连,这3根控制线是:

总线请求(BR)。该线有效表示至少有一个部件或设备要求使用总线。

总线忙(BS)。该线有效表示总线正在被某部件或设备使用。

总线批准(BG)。该线有效表示总线控制器响应总线请求。

与总线相连的所有部件经公共的BR线发出总线请求,只有在BS信号未建立前,BR才能被总线控制器响应,并送出BG回答信号。BG信号串行地通过每个部件,如果某个部件本身没有总线请求,则将该信号传给下一个部件;如果这个部件有总线请求,就停止传送BG信号,获得总线使用权。这时该部件将建立BS信号,并撤销总线请求信号BR,进行数据的传送。BS信号在数据传送完后撤销,BG信号也随之撤销。

链式查询的优点是只用很少几根线就能按一定的优先次序来实现总线控制,并很容易扩充。缺点是对查询链的故障很敏感,如果第i个部件中的查询链电路有故障,那么第i个以后的部件都不能工作。另外,因为查询的优先级是固定的,所以若优先级较高的部件出现频繁的总线请求时,优先级较低的部件就可能会"饿死"。

(2) 计数器定时查询方式

计数器定时查询方式采用一个计数器控制总线使用权。总线上的每个部件可以通过公共的BR线发出请求,总线控制器收到请求之后,在BS为"0"的情况下,计数器开始计数,计数值通过一组地址线发向各部件。当地址线上的计数值与请求总线部件的设备地址一致

时,该部件获得总线使用权,将BS线置"1",并中止计数,直至该部件完成数据传送之后,撤销BS信号。

这种计数可以从"0"开始,也可以从中止点开始。如果从"0"开始,各部件的优先次序和链式查询方式相同,优先级的次序是固定的。如果从中止点开始,即为循环优先级,各个部件使用总线的级别将相等。计数器的初始值还可以由程序来设置,这就可以方便地改变优先次序,增加系统的灵活性。

(3) 独立请求方式

在独立请求方式中,每一个共享总线的部件均有一对控制线:总线请求 BR_i 和总线批准 BG_i。当某个部件请求使用总线时,便发出 BR_i,总线控制器中有一个排队电路,根据一定的优先次序决定首先响应哪个部件的请求 BR_i,然后给该部件送回批准信号 BG_i。

独立请求方式的优点是响应快,然而这是以增加控制线数和硬件电路为代价的。此方式对优先次序的控制也是相当灵活的,它可以预先固定,也可以通过程序来改变优先次序。

总结上述3种方式,可见链式查询所需的控制线数最少;独立请求方式最多;计数器定时查询居中,对于 n 个部件的系统,共需要 $\lceil \log_2 n \rceil$ 根定时查询计数线。

5. 总线定时方式

在总线上通信的两个设备必须知道对方何时传送何种信息,因此双方需要有相应的通信协议,以确定如何交换信息,这就是定时方式的确定。最基本的定时方式有同步和异步两种,同步方式用一个公共的时钟信号对传输过程的每个步骤进行同步控制;异步方式用异步应答(握手)信号对传输过程的每个步骤进行定时控制。

同步总线的传输协议比较简单,但总线定时以最慢设备所花时间为标准,所以同步总线适合于存取时间相差不大的多个功能部件之间的通信,同时由于时钟偏移问题,导致同步总线不能过长。异步总线的传输协议称为"握手协议",只有当通信双方都同意时,才能进入到下一步。异步总线能够连接带宽范围很大的各种设备,总线能够加长而不用担心时钟偏移问题。

7.3 典型例题详解

【例7.1】 总线宽度的含义是什么?什么是总线的传输速率?某总线有104根信号线,其中数据总线(DB)32根,地址总线(AB)25根,控制总线(CB)47根,总线工作频率33MHz,问该总线的宽度是多少?其传输率是多少?

解:总线中数据总线的位数称为该总线的宽度。

总线的数据传输率为总线上每秒钟传输的最大字节数,单位是字节/秒(B/s)等。

由于本系统总线中数据总线为32位(b),所以有:

总线宽度 $W=32\text{b}$。

总线工作频率 $f=33\text{MHz}$。

数据传输率 $C=f\times W=33\text{MHz}\times\frac{32}{8}\text{B}=132\text{MB/s}$。

【例7.2】 假设总线的时钟频率为100MHz,总线的传输周期为4个时钟周期,总线的宽度为32位,试求总线的数据传输率。若想提高一倍数据传输率,可采取什么措施?

解：根据总线的时钟频率为100MHz，可得：

1个时钟周期为1÷100MHz=0.01μs。

一个总线传输周期等于4个时钟周期，0.01μs×4=0.04μs。

总线的宽度为32b，等于4B。

故总线的数据传输率为4B÷(0.04μs)=100MB/s。

也可以根据下述公式计算：

总线数据传输率=总线宽度×总线频率

式中的总线频率等于总线时钟频率除以每个总线周期的时钟数。本例中，

总线数据传输率=4B×100MHz÷4=100MB/s。

若想提高一倍数据传输率，有两种方法：① 在不改变时钟频率的前提下，将数据线的宽度改为64位；② 仍保持数据宽度为32位，但使总线的时钟频率增加到200MHz。

【例7.3】 有4个部件A、B、C、D，其响应优先权为A>B>C>D，分别画出链式查询、计数器定时查询和独立请求排队电路，以及与总线控制器之间的连接关系。

解：链式查询电路及与总线控制器的连接如图7-3所示。若BR_B有请求，则BR=1，总线控制器检查总线忙否，若总线不忙，则立即发总线批准信号BG，因为$BR_A=0$，所以$BS_A=0$，将BG信号传到下一个部件。这时由于$BR_B=1$，所以$BS_B=1$，部件B得到总线使用权，同时BG信号被截住，不再传下去，从而封锁了后面部件的请求。

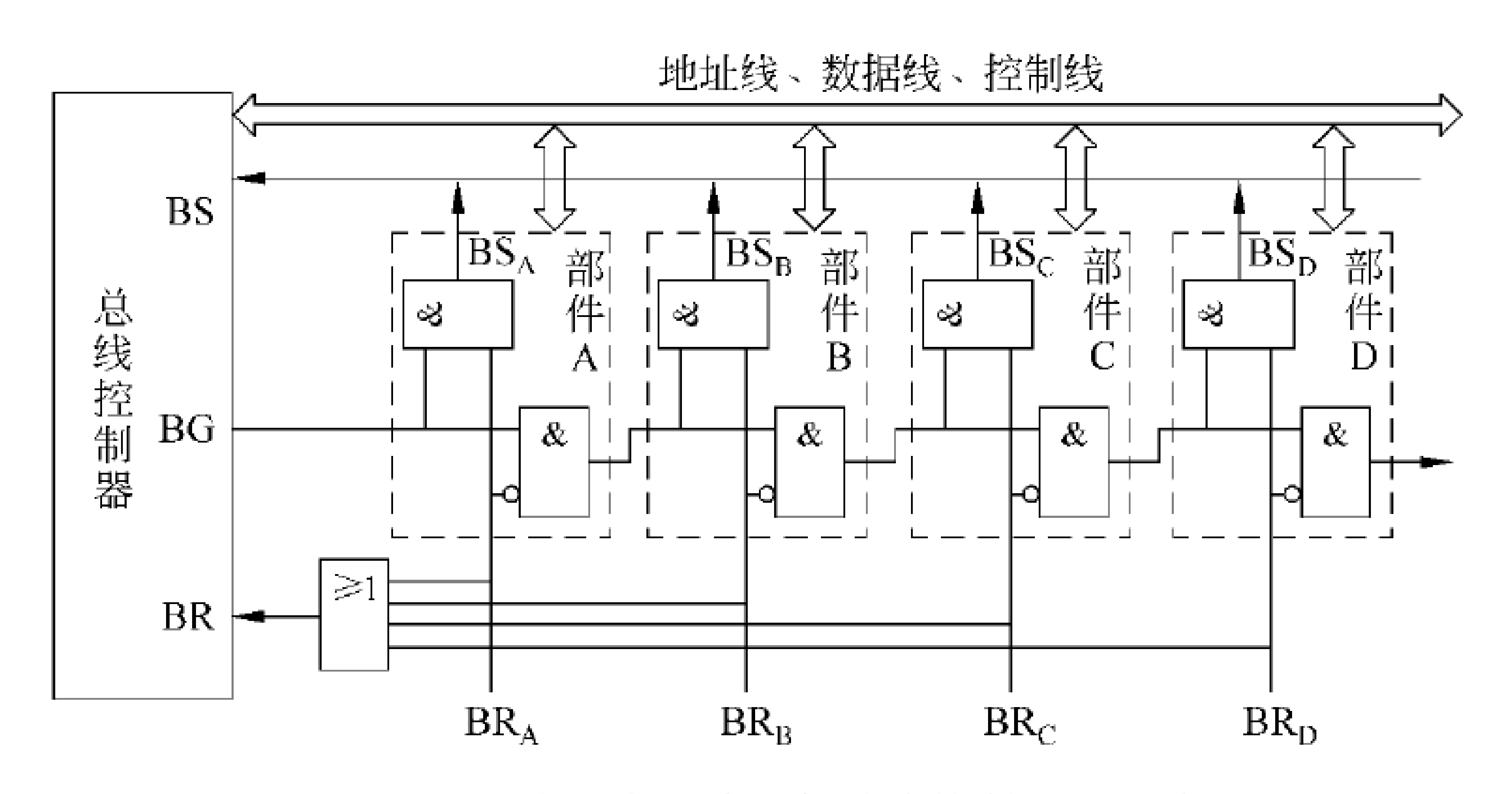


图7-3　链式查询电路及与总线控制器的连接

计数器定时查询电路及与总线控制器的连接如图7-4所示。若BR_B有请求，则BR=1，总线控制器检查总线忙否，若总线不忙，计数器开始计数，设计数器初值为00。由于$BR_A=0$，所以$BS_A=0$，计数器继续计数到01，因为$BR_B=1$，所以$BS_B=1$，部件B获得总线使用权，计数器停止计数。

独立请求方式与排队电路如图7-5所示。当总线上的部件需要使用总线时，经各自的总线请求线发送总线请求信号，在总线控制器中排队，当总线控制器按一定优先次序决定批准某个部件的请求时，则给该部件发送总线批准信号，该部件接到此信号就获得总线使用权。

*【例7.4】 假设某系统总线在一个总线周期中并行传输4B信息，一个总线周期占用2个时钟周期，总线时钟频率为10MHz，则总线带宽是________。

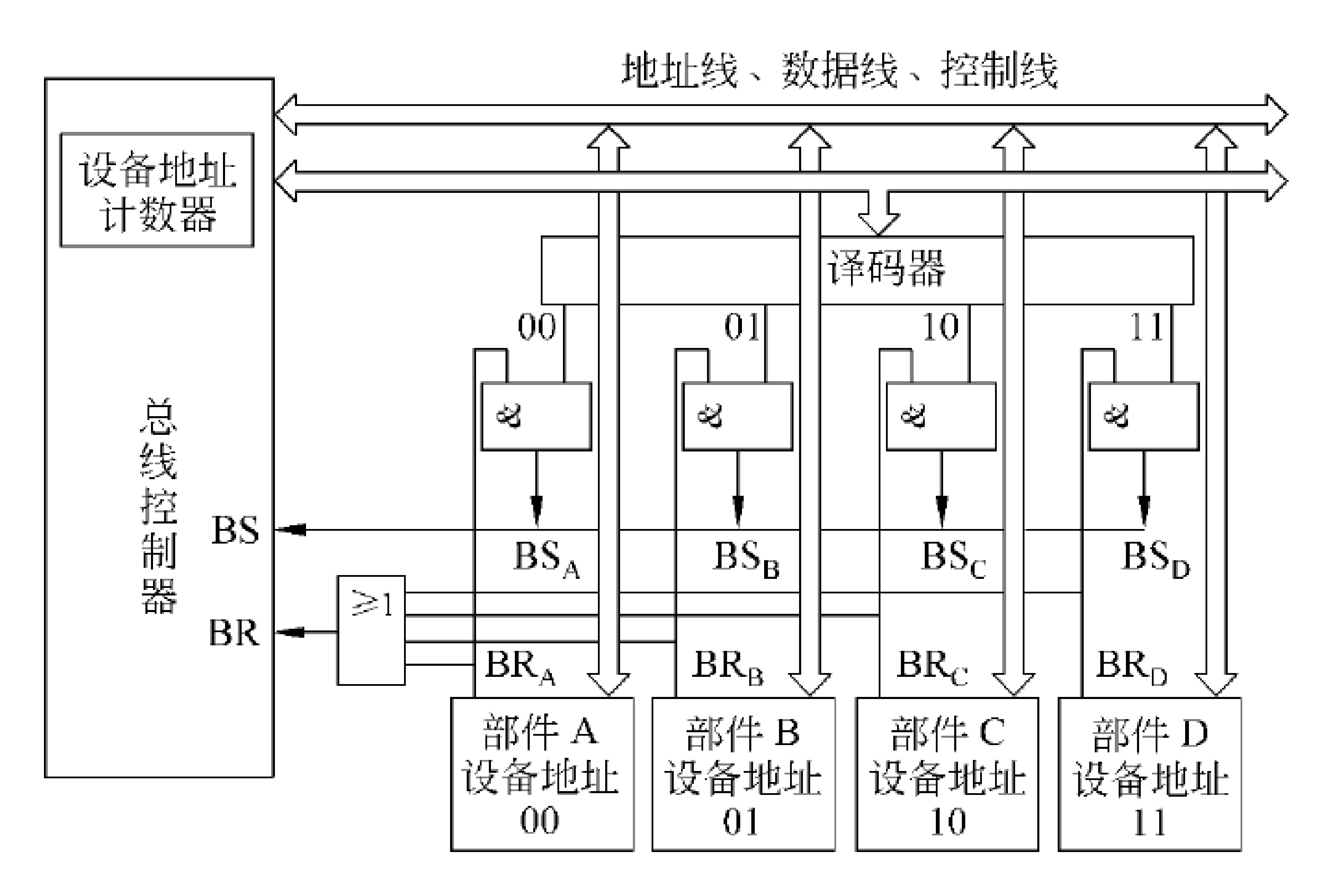


图 7-4 计数器定时电路及与控制器的连接

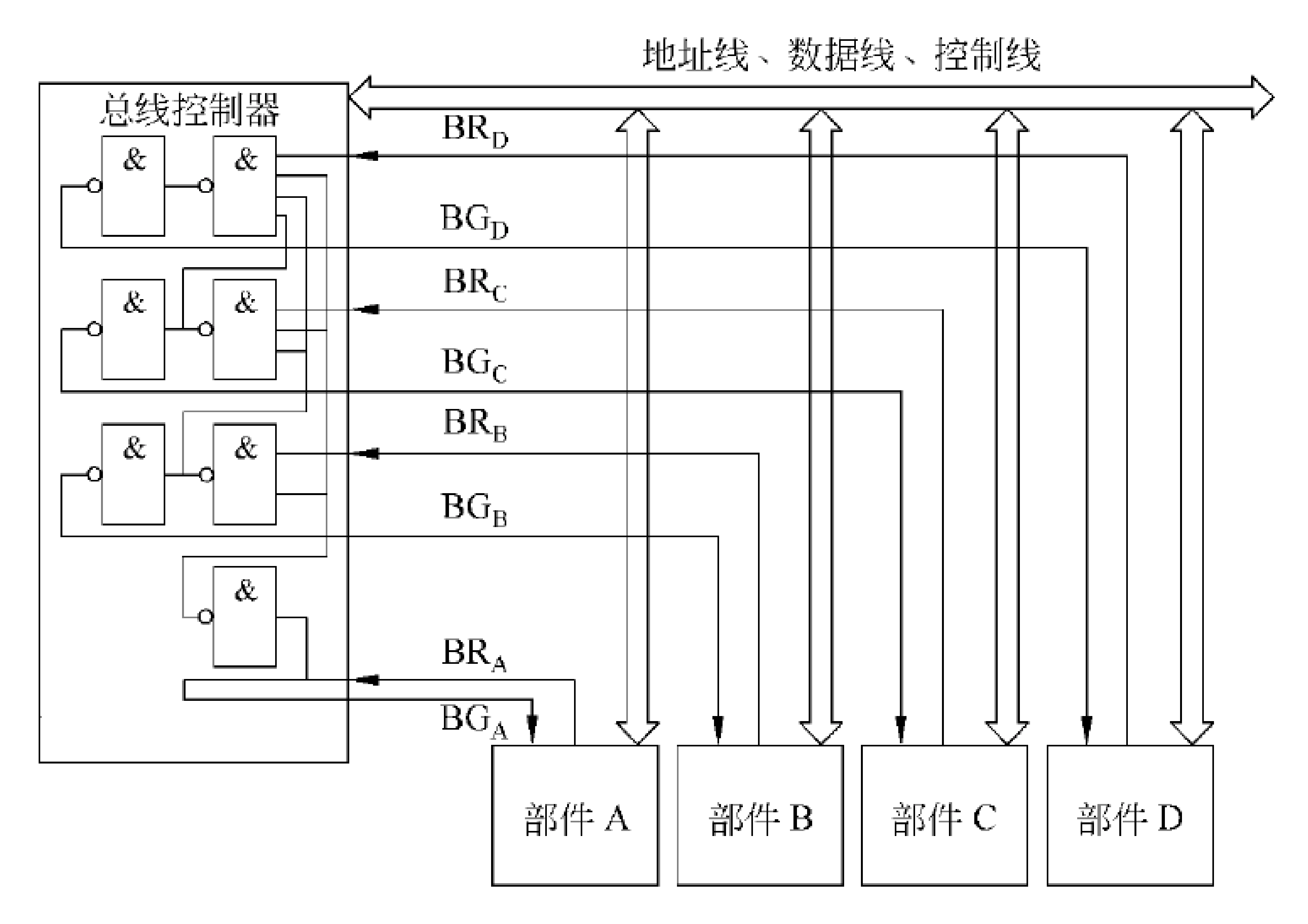


图 7-5 独立请求方式与排队电路

A. 10MB/s　　B. 20MB/s　　C. 40MB/s　　D. 80MB/s

解：B。

分析：因为一个总线周期占用2个时钟周期，完成一个32位数据的传送。总线时钟频率为10MHz，时钟周期为0.1μs，总线周期占用2个时钟周期，为0.2μs。一个总线周期中并行传输4B信息，则总线带宽是4B÷0.2μs=20MB/s。

*【例7.5】 下列选项中的英文缩写均为总线标准的是________。

A. PCI、CRT、USB、EISA　　B. ISA、CPI、VESA、EISA

C. ISA、SCSI、RAM、MIPS　　D. ISA、EISA、PCI、PCI-Express

解：D。

分析：选项A、选项B、选项C中均有不是总线标准的英文缩写，如CRT、USB、CPI、RAM、MIPS等，只有选项D中的英文缩写均为总线标准。

*【例7.6】 在系统总线的数据线上，不可能传输的是________。

A. 指令　　B. 操作数
C. 握手(应答)信号　　D. 中断类型码

解：C。

分析：握手(应答)信号属于通信联络控制信号，不可能在数据总线上传输。而指令、操作数和中断类型码都可以在数据线上传输。

*【例 7.7】 某同步总线的时钟频率为 100MHz，宽度为 32 位，地址/数据线复用，每传输一个地址或数据占用一个时钟周期。若该总线支持突发(猝发)传输方式，则一次“主存写”总线事务传输 128 位数据所需要的时间至少是________。

A. 20ns　　B. 40ns　　C. 50ns　　D. 80ns

解：C。

分析：总线的时钟频率为 100MHz，则时钟周期为 10ns。传输一个 128 位的数据至少需要 5 个时钟周期，其中一个时钟周期传输地址，4 个时钟周期传输数据，所以至少需要 50ns。

*【例 7.8】 下列关于 USB 总线特性的描述中，错误的是________。

A. 可实现外设的即插即用和热插拔　　B. 可通过级联方式连接多台外设
C. 是一种通信总线，可连接不同外设　　D. 同时可传输 2 位数据，数据传输率高

解：D。

分析：USB(通用串行总线)是一种外设总线标准，具有即插即用功能，并支持热插拔。USB 在一台计算机上最多可以同时支持 127 台设备的运行。

*【例 7.9】 某同步总线采用数据线和地址线复用方式，其中地址/数据线有 32 根，总线时钟频率为 66MHz，每个时钟周期传送两次数据(上升沿和下降沿各传送一次数据)，该总线的最大数据传输率(总线带宽)是________。

A. 132MB/s　　B. 264MB/s　　C. 528MB/s　　D. 1056MB/s

解：C。

分析：总线时钟频率为 66MHz，每个时钟周期传送两次数据，总线数据传输率＝总线宽度×总线频率×2＝4B×66MHz×2＝528MB/s。

*【例 7.10】 一次总线事务中，主设备只需给出一个首地址，从设备就能从首地址开始的若干连续单元读出或写入多个数据。这种总线事务方式称为________。

A. 并行传输　　B. 串行传输　　C. 突发传输　　D. 同步传输

解：C。

分析：突发传送事务由一个地址阶段和多个数据阶段构成，用于传送多个连续单元的数据，地址阶段送出的是连续区域的首地址。因此，一次突发传送事务可以传送多个数据。

7.4 同步测试习题及解答

7.4.1 同步测试习题

一、填空题

1. 三态门电路比普通门电路多一种________状态。

2. 总线的________裁决方式速度最高。

二、选择题

1. 系统总线中,划分数据线、地址线和控制线的根据是________。

A. 总线所处的位置　　B. 总线的传输方向

C. 总线的传输内容　　D. 总线的控制方式

2. 系统总线中地址线的作用是________。

A. 用于选择主存单元

B. 用于选择进行信息传输的设备

C. 用于指定主存单元和I/O设备接口电路的地址

D. 用于传送主存物理地址和逻辑地址

3. 挂接在总线上的多个部件________。

A. 只能分时向总线发送数据,并只能分时从总线接收数据

B. 只能分时向总线发送数据,但可同时从总线接收数据

C. 可同时向总线发送数据,并同时从总线接收数据

D. 可同时向总线发送数据,但只能分时从总线接收数据

4. 总线的从设备指的是________。

A. 申请作为从设备的设备　　B. 被主设备访问的设备

C. 掌握总线控制权的设备　　D. 总线源设备

5. "总线忙"信号的建立者是________。

A. 获得总线控制权的设备　　B. 发出"总线请求"信号的设备

C. 总线控制器　　D. CPU

6. 在集中式总线控制中,________方式响应时间最快。

A. 链式查询　　B. 计数器定时查询

C. 独立请求　　D. 不能确定哪一种

7. 在计数器定时查询方式下,正确的描述是________。

A. 总线设备的优先级可变　　B. 越靠近控制器的设备优先级越高

C. 各设备的优先级相等　　D. 各设备获得总线使用权的机会均等

8. 为了对 n 个设备使用总线的请求进行仲裁,在独立请求方式中需要使用的控制线数量为________。

A. n　　B. 3　　C. $2+\lceil \log_2 n \rceil$　　D. $2n+1$

9. 在计数器定时查询方式下,若每次计数从上一次计数的中止点开始,则________。

A. 设备号小的优先级高　　B. 每个设备使用总线的机会相等

C. 设备号大的优先级高　　D. 每个设备的优先级相等

三、判断题

1. 微型机中的系统总线包括数据总线、地址总线和控制总线,所以称它为三总线。( )

2. 一个总线在某一时刻可以有多对主、从设备进行通信。( )

四、简答题

1. 为什么要有总线判优控制?

2. 试说明计数器定时查询方式的优缺点。

五、综合题

某总线时钟频率为66MHz,在一个64位总线中,总线数据传输的周期是7个时钟周期传输6个字的数据块。

(1) 问总线的数据传输率是多少?

(2) 如果不改变数据块的大小,而是将时钟频率减半,问这时总线的数据传输率是多少?

7.4.2 同步测试习题解答

一、填空题

1. 浮空。

2. 独立请求。

二、选择题

1. C。系统总线按传送信息的不同又可以细分为地址总线、数据总线和控制总线。

2. C。地址总线主要用来指出数据总线上的数据在主存单元的地址或I/O设备的地址。因为对于单总线(系统总线)结构,主存和I/O设备都挂在总线上。

3. B。为了使总线上的数据不发生“冲突”,挂接在总线上的多个设备只能分时地向总线发送数据,即每个时刻只能有一个设备向总线传送数据,而从总线接收数据的设备可有多个,因为接收数据的设备不会对总线产生“干扰”。

4. B。总线设备可分为主设备和从设备,掌握总线控制权的设备是总线主设备,而被主设备访问的设备是从设备。

5. A。只有申请使用总线并获得总线控制权的设备,才能发出“总线忙”信号。

6. C。独立请求方式的响应时间最快,然而这是以增加控制线数和硬件电路为代价的。

7. A。在计数器定时查询方式下,根据计数值的初始值的不同,总线设备的优先级是可变的。如果计数值从“0”开始,离总线控制器最近的设备具有最高的优先级。如果计数值从上一次的中止点开始,即为循环优先级,各个部件使用总线的机会将相等。计数器的初始值还可以由程序来设置,这样就可以更方便地改变优先级。

8. D。对于n个设备而言,链式查询方式需要3条控制线,计数器定时查询方式需要$2+\lceil \log_2 n \rceil$条控制线,而独立请求方式需要$2n+1$条控制线,包括n条总线请求线、n条总线批准线和1条总线忙线。

9. B。如果每次计数从上一次计数的中止点开始,即为循环优先级,各个部件使用总线的机会将相等。

三、判断题

1. ×。微型机中的系统总线称为单总线。

2. ×。在一个总线传输周期内,总线上只能有一个主控设备控制总线,选择一个从设备与之进行通信,或若对所有其他设备进行广播。

四、简答题

1. 总线的特性是分时共享,同时可以有多个设备连接在同一个总线上,但每一时刻总线只能完成一对设备之间的信息传送。当有多个设备同时要使用总线传输信息时,需要通

过总线判优机制，在多个请求中选择一个，让其控制总线来传输信息，其他设备则暂时等待并在以后的判优中再逐一被选中。

2. 计数器定时查询方式的优点是可以方便地改变优先次序，增加系统的灵活性；缺点是控制线数较多(需 $2+\lceil \log_2 n \rceil$ 根)，扩展性稍差，控制较复杂。

五、综合题

(1) 48B×66MHz÷7=452.6MB/s。

(2) 48B×33MHz÷7=226.3MB/s。

第 8 章 外部设备

8.1 基本内容摘要

- 外部设备概述
  - 外部设备的分类
- 磁介质存储器的性能和原理
  - 磁介质存储器的读写
  - 磁介质存储器的技术指标
  - 数字磁记录方式
  - 编码方式的比较
- 磁介质存储设备
  - 硬盘存储器的基本结构与分类
  - 硬盘驱动器
  - 硬盘的信息分布和磁盘地址
  - 硬盘存储器的技术参数
  - 硬盘的分区域记录
  - 软磁盘存储器
- 磁盘阵列
- 光盘存储器
  - 光盘存储器的类型
  - 光盘存储器的组成及工作原理
- 新型辅助存储器
  - 基于磁或磁光介质的可移动存储器
  - 基于电子器件的存储器
- 键盘输入设备
  - 键开关与键盘类型
  - 键盘扫描
  - 微型机键盘
- 其他输入设备
- 打印输出设备

- ◆ 打印机概述
- ◆ 打印机的主要性能指标
- ◆ 针式打印机工作原理
- ◆ 喷墨打印机工作原理
- ◆ 激光打印机工作原理

● 显示设备

- ◆ 显示器概述
- ◆ CRT 显示器
- ◆ 字符显示器的工作原理
- ◆ 图形显示器的工作原理

8.2 重点难点梳理

1. 磁介质存储器

磁介质存储器是一种可兼作输入和输出用的外部设备。它可接收从主机输出的信息，并存储起来，也可以向主机输入事先存储好的信息。就其存储功能而言，相对于主机内部的主存储器，被称为辅助存储器或外存储器。

在磁介质存储器中，信息被记录在磁层上。磁层是指一薄层的磁性材料，将它均匀地涂抹在圆形的铝合金和塑料的载体上就成为磁盘，涂抹在聚酯塑料带上就成为磁带。

磁头是磁介质存储器用来实现“电-磁”转换的重要元件。写磁头能把电脉冲表示的二进制代码转换成磁记录介质上的磁化状态，即实现电→磁转换；读磁头能把磁记录介质上的磁化状态转换成电脉冲，即实现磁→电转换。

2. 磁介质存储器的记录密度

记录密度又称存储密度，是指磁介质存储器上单位长度或单位面积所存储的二进制信息量。通常以道密度和位密度表示，也可用两者的乘积——面密度来表示。

道密度又称横向密度，是指垂直于磁道方向上单位长度中的磁道数目，道密度的单位通常用道/in 或者道/cm 表示。

位密度又称纵向密度，是指沿磁道方向上单位长度中所记录的二进制信息的位数，位密度的单位通常用 b/in 或者 b/cm 表示。

传统磁盘驱动器的位密度是变化的。因为内圈磁道的周长短，外圈磁道的周长长，所以内圈磁道的位密度高，外圈磁道的位密度低，最内圈磁道的位密度(最大位密度)决定了磁盘驱动器的容量。而在当今采用了分区域记录技术的 IDE 和 SCSI 驱动器上，位密度将不再变化。

3. 平均存取时间和数据传输率

在磁介质存储器中，当磁头接到读写命令，从原来的位置移动到指定位置，并完成读写操作的时间称为存取时间。对于采用顺序存取方式的多道并行读写的磁带存储器，没有寻找磁道的问题，故只需要考虑磁头等待记录块的等待时间和信息的读写操作时间。对于采用直接存取方式的磁盘存储器，存取时间主要包括 4 部分：第一部分是指磁头从原先位置移动到目的磁道所需要的时间，称为定位时间或寻道时间；第二部分是指在到达目的磁道以

后，等待被访问的记录块旋转到磁头下方的等待时间，称为旋转时间或等待时间；第三部分是信息的读写操作时间，也称为传输时间；第四部分是磁盘控制器的开销。由于寻找不同磁道和等待不同记录块所花费的时间不同，所以通常取它们的平均值。传输时间和控制器的开销相对平均寻道时间 T_s 和平均等待时间 T_w 来说可以忽略不计，所以磁盘的平均存取时间 T_a 可以近似等于：

$$T_a \approx T_s + T_w = \frac{t_{smix} + t_{smax}}{2} + \frac{t_{wmix} + t_{wmax}}{2}$$

其中，T_s 等于磁头从 0 号磁道移动到最末一个磁道所需时间的一半；T_w 等于磁盘旋转一圈所需时间的一半。

数据传输率是指磁介质存储器在单位时间内向主机传送数据的字节数或位数。如果磁盘的旋转速度为 r 转/s，每条磁道的容量为 N 个 B，则数据传输率 $D_r = rN$B/s。

4. 常见的几种磁记录方式与自同步能力

(1) 不归零制(NRZ)：记录“1”时，磁头线圈中通以正向电流；记录“0”时，通以反向电流。只有当记录的相邻两位信息不相同(即“0”和“1”交替)时，写电流才改变方向，称为见变就翻的不归零制。

(2) 不归零—1 制(NRZ-1)：记录“1”时，磁头线圈中写电流改变方向，使磁层磁化翻转；而记录“0”时，写电流方向维持不变，保持原来的磁化状态，所以称为见 1 就翻的不归零制。

(3) 调相制(PE)：记录“1”时，写电流在位周期中间由负变正；记录“0”时，写电流在位周期中间由正变负。

(4) 调频制(FM)：记录“1”时，写电流在位周期中间和边界各改变一次方向，对应的磁层有两次磁化翻转；记录“0”时，写电流仅在位周期边界改变一次方向，对应的磁层只有一次磁化翻转。

(5) 改进的调频制(MFM)：记录“1”时，写电流在位周期中间改变方向，产生磁化翻转；记录独立的一个“0”，写电流不改变方向，不产生磁化翻转；记录连续的两个“0”，写电流在位周期边界改变方向，产生磁化翻转。

近年来发展的高密度磁盘主要选用游程长度受限码(RLL)。这是一种提高记录密度，增强编码抗干扰能力的方法。它是将数据序列中的数据位几位分成一组，然后按一定的变换规则变换成对应的记录码，再采用 NRZ-1 制写入记录介质。

自同步能力是指能否从单个磁道读出的脉冲序列中提取同步时钟脉冲的能力。NRZ 制和 NRZ-1 制无自同步能力，其余的记录方式均有自同步能力。具有自同步能力的记录方式的自同步能力也有强有弱，强者提取同步信号的电路比较简单，比较容易实现。自同步能力的强弱可以用最小磁化翻转间隔和最大磁化翻转间隔的比值 R 来衡量。R 值越大，自同步能力越强。

5. 硬盘存储器的信息分布

硬盘中的信息是按记录面、圆柱面、磁道、扇区的层次安排的。

(1) 记录面。一台硬盘驱动器中有多个盘片，每个盘片有两个记录面，每个记录面对应一个磁头，所以记录面号就是磁头号。

(2) 磁道。在记录面上，一条条磁道形成一组同心圆，最外圈的磁道为 0 号，往内则磁

道号逐步增加。

(3) 圆柱面。在一个盘组中,各记录面上相同编号(位置)的诸磁道构成一个圆柱面。例如,某驱动器有4片8面,则8个0号磁道构成0号圆柱面,8个1号磁道构成1号圆柱面……硬盘的圆柱面数就等于一个记录面上的磁道数,圆柱面号即对应的磁道号。引入圆柱面的概念,是为了提高硬盘的存储速度。当主机要存入一个较长的文件时,若一条磁道存不完,应首先将其尽可能地存放在同一圆柱面中。如果仍存放不完,再存入相邻的圆柱面内。

(4) 扇区。一条磁道被划分为若干个段,每个段称为一个扇区或扇段,每个扇区存放一个定长信息块(如512B)。

6. 磁盘地址

主存储器与磁盘存储器之间交换信息是以扇区为单位的,所以在访问磁盘存储器时,需要提供的磁盘地址包括驱动器号(台号)、圆柱面(磁道)号、记录面(磁头)号、扇区号。通常,主机通过一个磁盘控制器可以连接几台磁盘驱动器,所以必须首先送出驱动器号,以选中指定的磁盘驱动器;然后根据指定的磁道号进行寻道定位操作;接下来根据磁头号确定指定的盘面;最后由扇区号找到指定的扇区进行读写操作。

7. 编码键盘和非编码键盘

编码键盘是用硬件电路来识别按键代码的键盘,当某一键按下后,相应电路即给出一组编码信息(如ASCII码)送主机去进行识别及处理。编码键盘的响应速度快,但它以复杂的硬件结构为代价,并且其硬件的复杂程度随着键数的增加而增加。

非编码键盘是用较为简单的硬件和专门的键盘扫描程序来识别按键的位置,即当按某键以后并不给出相应的ASCII码,而提供与按下键相对应的中间代码(如扫描码),然后再把中间代码转换成对应的ASCII码。非编码键盘的响应速度不如编码键盘,但是它通过软件编程可为键盘中某些键的重新定义提供更大的灵活性,因此得到了广泛使用。

8. 非编码键盘的键盘扫描方法

键开关被排列成 M 行 $\times N$ 列的矩阵结构,每个键开关位于行和列的交叉处。通过执行键盘扫描程序对键盘矩阵进行扫描,以识别被按键的行、列位置。键盘扫描方法有逐行扫描法和行列扫描法两种。

1) 逐行扫描法

逐行扫描法处理的步骤如下:

(1) 首先由CPU对输出端口(行线)的各位置"0",即将各行全部接地,然后CPU再从输入端口(列线)读入数据。若读入的数据全为"1",表示无键按下;只要读入的数据中有一个不为"1",表示有键按下。

(2) 接着要查出按键的位置。CPU首先使 $X_0=0$,$X_1 \sim X_7$ 全为"1",读入 $Y_0 \sim Y_7$,若全为"1",表示按键不在这一行;接着使 $X_1=0$,其余各位全为"1",读入 $Y_0 \sim Y_7$……直至 $Y_0 \sim Y_7$ 不全为"1"为止,从而确定了当前按下的键在键盘矩阵中的位置。

(3) 得到的行号和列号表示按下键的位置码。

2) 行列扫描法

在扫描行时,读列线,若读得的结果全为"1",说明没有键按下,即尚未扫描到闭合键;若某一列为低电平,说明有键按下。然后扫描列线,读行线,即可确定闭合键所在位置的行号

和列号，即得到闭合键的行列扫描码。

9. 文本模式和图形模式打印机

1）文本模式

在文本模式中，主机向打印机输出字符代码（ASCII 码）或汉字代码，打印机则依据代码从位于打印机上的字符库或汉字库中取出点阵数据，在纸上“打”出相应字符或汉字。与图形模式相比，文本模式所需传送的数据量少，占用主机 CPU 的时间少，因而效率较高，但所能打印的字符或汉字的数量受到字库的限制。

2）图形模式

在图形模式中，主机向打印机直接输出点阵图形数据，有一个“1”就“打”一个点。在这种模式下，CPU 能灵活地控制打印机输出任意图形，从而可打印出字符、汉字、图形、图像等。但是图形模式所需传送的数据量大，占用主机大量的时间。例如，打印一个 24×24 点阵的汉字，传送字符点阵图形的数据量（72B）远大于传送字符代码时的数据量（2B）。

10. 针式打印机的工作原理

针式打印机的打印控制系统是一个专门的微处理器系统。微处理器通过执行存放在 ROM 中的控制程序来实现与主机的数据通信和控制打印机的各种动作。以文本模式为例，RAM 为打印缓冲区，存放主机送来的 ASCII 码或汉字代码，其容量可存放一行打印数据。ROM 字库用来存放各个 ASCII 码的点阵编码或汉字的点阵码。

主机要输出打印信息时，首先要检查打印机所处的状态。当打印机空闲时，允许主机发送字符代码或汉字代码。打印机的微处理器接收从主机送来的字符代码或汉字代码后，先判断它们是可打印的符号还是只执行某种控制操作的控制字符（如“回车”、“换行”等）。如果是可打印的符号就将其代码送入打印行缓冲区（RAM）中，接口电路产生回答信息，通知主机发送下一个字符代码或汉字代码。如此重复，把要打印的一行符号的代码都存入数据缓冲区。当缓冲区接收满一行打印的字符代码或汉字代码后，停止接收，转入打印。

打印时，首先从 ROM 字库中寻找到与字符和汉字相对应的点阵首列地址，然后按顺序一列一列地找出字符和汉字的点阵，送往打印头控制驱动电路，激励打印头出针打印。一个字符或汉字打印完，字车移动几列，再继续打印下一个字符或汉字。一行字符或汉字打印完后，请求主机送来第二行打印字符代码或汉字代码，同时输纸机构使打印纸移动一个行距。

打印字库 ROM 中存放着打印字符和汉字的列点阵码，为节省字库的空间，可将非打印符号去掉。

打印 ASCII 码字符时，将 ASCII 码的前 32 个字符去掉（因为前 32 个字符为非打印字符），根据 ASCII 在字库中寻找点阵码的公式为：

字符的列点阵首地址＝（ASCII 码－20H）×列点阵数＋字库首地址

例如，某字模的点阵为 5×7（横向 5 点，纵向 7 点），字库首地址为 100H，则字符“A”（ASCII 码为 41H）在字库中的首地址为：

（41H－20H）×5＋100H＝205H

由于每个字符有 5 个列点阵码，此时列计数器为 3 位，某字符的列点阵码在字库中的位置如下：

地址高位＝（ASCII 码－20H），地址低位＝列计数值

如上例中字符“A”在字库中的点阵处于位置（未考虑字符首地址）为 0100001 000～

0100001 100,即108H～10CH。

打印汉字时,根据汉字机内码在字库中寻找点阵码的公式为:

汉字的列点阵首地址=(汉字机内码－第一个汉字的机内码)
×一个汉字点阵码存放的字节数＋汉字库首地址

11. CRT显示器的光栅扫描

在光栅扫描方式中,电子束在水平和垂直同步信号的控制下,有规律地扫描整个屏幕。扫描的方法如下:电子束从显示屏的左上角开始,沿水平方向从左向右扫描,到达屏幕右端后迅速水平回扫到左端下一行位置,又从左到右匀速地扫描。这样一行一行地扫描,直到屏幕的右下角,然后又垂直回扫,返回屏幕左上角,重复前面的扫描过程。在水平和垂直回扫时,电子束是"消隐"的,荧光屏上没有亮光显示。这样,在CRT的屏幕上形成了一条条水平扫描线,称为光栅。

12. 显示器的显示模式

显示模式分为字符模式和图形模式。

在字符模式下,显示缓冲区中存放着显示字符的代码(ASCII码)和属性。显示屏幕划分为若干个字符显示行和列,例如80列×25行。

图形模式对所有点均可寻址,常称为位图化的显示器,因为屏幕上的每个像素都对应显示缓冲区中的一位或多位。

13. 显示缓冲区(VRAM)

为了不断地提供刷新画面的信号,必须把字符或图形信息存储在一个显示缓冲区中,这个缓冲区又称为视频存储器(VRAM)。显示器一方面对屏幕进行光栅扫描;另一方面同步地从VRAM中读取显示内容,送往显示器件。因此,对VRAM的操作是显示器工作的软、硬件界面所在。

VRAM的容量由分辨率和灰度级决定,分辨率越高,灰度级越高,VRAM的容量就越大。同时,VRAM的存取周期必须满足刷新频率的要求。

字符模式的VRAM通常分成两部分:字符代码缓存和显示属性缓存。字符代码缓存中存放着显示字符的ASCII码,每个字符占1B;显示属性缓存中存放着字符的显示属性,一般也占1B。VRAM的最小容量是由屏幕上字符显示的行、列规格来决定的。例如,一帧字符的显示规格为80×25,那么VRAM中的字符代码缓存的最小容量就是2KB。缓存的容量也可以大于一帧字符数,用来同时存放几帧字符的代码。在这种情况下,通过控制缓存的指针就可以在屏幕上显示不同帧中的字符内容,实现屏幕的硬件滚动。

字符模式VRAM的地址和屏幕上显示该字符的位置相对应。设字符在屏幕上的位置坐标为(X,Y),即行地址为X,列地址为Y,则字符代码所在的存储地址=$(X\times80+Y)\times2$;显示属性所在的存储地址=$(X\times80+Y)\times2+1$。

图形模式的显示信息以二进制的形式存储在VRAM中,这些信息是图形元素的矩阵数组,在最简单的情况下,只需要存储两值图形,即用"0"表示黑色(暗点),用"1"表示白色(亮点)。用VRAM的1位表示1个点,所以VRAM的1个字节可以存放8个点。例如,一个CRT显示器的分辨率为640×200,在无灰度级的单色显示器中,只需要16KB的VRAM。在彩色显示或单色多灰度显示时,每个点需要若干位来表示。例如,若用2位二进制代码表示1个点,那么每个点便能选择显示4种颜色,但是此时VRAM的1个字节只

能存放4个点，如果显示器的分辨率不变，VRAM的容量就要增加一倍。反之，若VRAM容量一定，随着分辨率的增高，显示的颜色数将减少。所以在图形模式下，对VRAM的需求随显示分辨率的大小和颜色数的多少而不同：

VRAM的容量=分辨率×颜色深度

颜色深度与颜色数的对应关系为：

$$颜色深度=\log_2 颜色数$$

14. 字符显示器的工作原理

VRAM中存放的是字符的ASCII码，不是点阵信息。若要显示出字符的形状，还要有字符发生器(字符库)的支持。

字符发生器中存放字符的行点阵码，字符发生器的地址由两部分组成：字符的行点阵码首地址高位=ASCII码-20H，字符的行点阵码首地址低位=行计数值。

例如，某字模的点阵为7×9(横向7个点，纵向9个点)，即行计数器为4位，则字符"A"(ASCII码为41H)，41H-20H=21H，其在字符发生器中的点阵处于的位置为0100001 0000～0100001 1000，即210H～218H。

在屏幕上，每个字符行一般要显示多个字符，而电子束在进行光栅扫描时，是沿屏幕从左向右的方向扫描完第一行，再扫描第二行。按照这种扫描方式，在显示字符时，并不是对一排的每个字符单独进行点阵扫描(即扫描完一个字符的各行点阵，再扫描同排另一个字符的各行点阵)，而是采用对同一排的所有字符的点阵进行逐行依次扫描。例如，某字符行欲显示的字符是A、B、C……T，显示电路首先根据各字符代码依次从字符发生器取出A、B、C……T各个字符的第一行点阵代码，并且在字符行第一条扫描线位置上显示出这些字符的第一行点阵；然后再依次取出该排各个字符的第二行代码，并且在屏幕上扫出它们的第二行点阵。如此循环，直到扫描完该字符行的全部扫描线，那么每个字符的所有点阵(如9行点阵)便全部显示在相应的位置上，屏幕上就出现了一排完整的字符。当显示下一排字符时，重复上述的扫描过程。

15. 字符显示器的控制逻辑电路

光栅扫描显示是以行扫描线为单位，在屏幕范围内逐行扫描、逐行显示。

字符显示器的控制逻辑电路由时钟、一组计数器和一些门电路组成，显示控制逻辑如图8-1所示。这些计数器有：

- 点计数器：时序脉冲送移位寄存器，控制并/串转换。归零输出脉冲送字计数器，作为计数脉冲。
- 字计数器：该计数器用来控制一排中的第n个字符。它的输出送到VRAM，作为其X向地址。归零输出作为行计数器计数脉冲。
- 行计数器：该计数器控制一排字符的第n行，其输出送字符发生器，作为读取字符发生器的低位地址。归零输出作为排计数器的输入。
- 排计数器：该计数器控制显示第n排字符，其输出送到VRAM，作为其Y向地址。

假定点振荡器的输出直接用于点计数器的计数脉冲，则其频率为：

$$f=s\times q_1\times q_2\times q_3\times q_4$$

式中，s为画面刷新频率，q_1、q_2、q_3、q_4分别为点、字、行、排计数器的最大值。

16. 图形显示器的工作原理

若彩色图形显示器的分辨率为640×480，可以同时显示16种颜色。VRAM中存放着

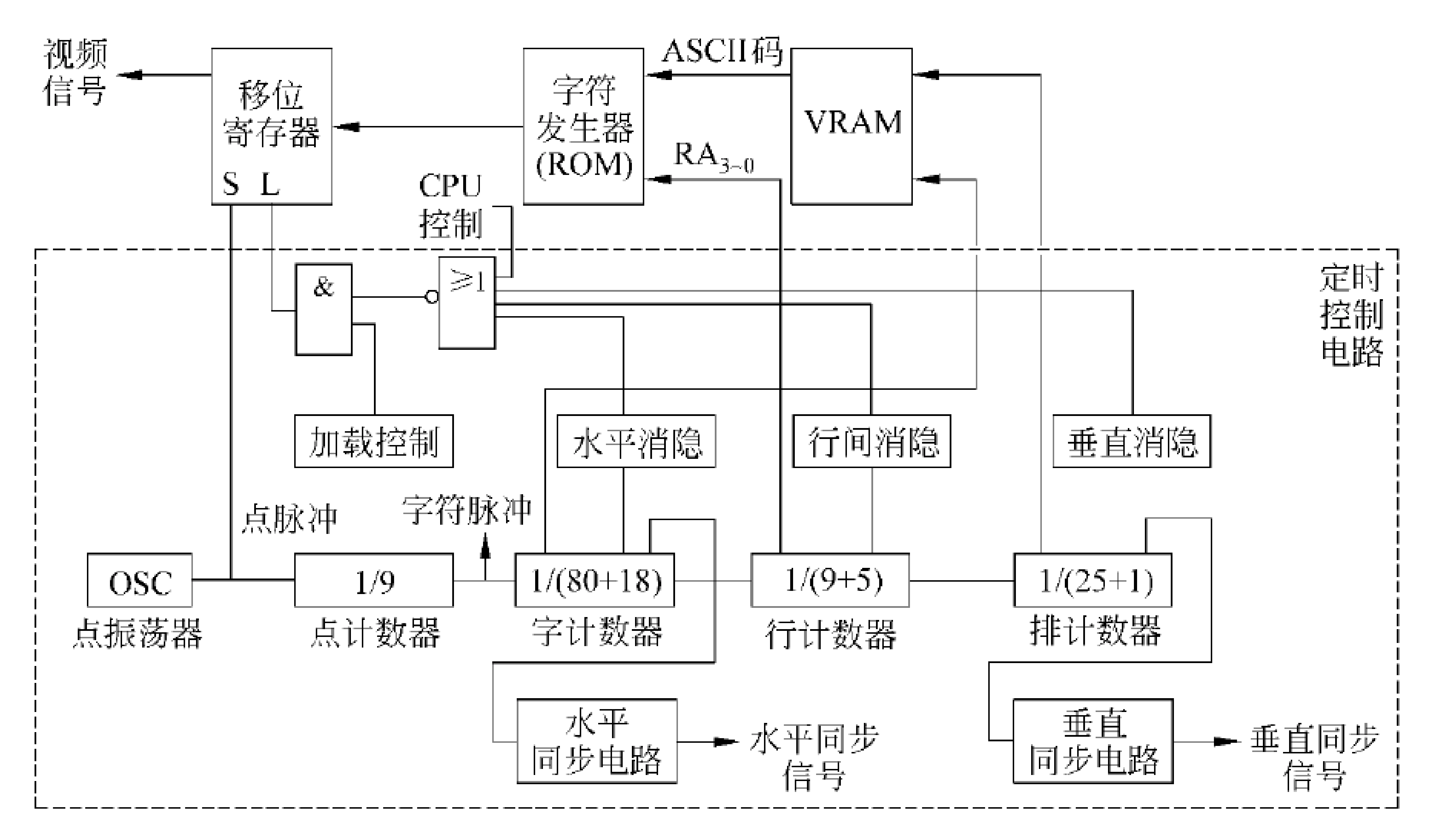


图 8-1　字符显示器控制逻辑电路

显示的图形点阵数据，由于计算机只能以二进制方式存放数据，每位只有两种状态“0”或“1”。对于单色显示，VRAM 中的每一位对应画面上的一个像素点，该位为“1”即表示画面上的这一点是亮点。而对于彩色显示(如 16 种颜色)，就需要用 VRAM 中的 4 位来定义一种颜色。在彩色图形显示器中经常采用彩色位平面的存储结构来表示颜色信息。每个彩色位平面由单一位组成，并表示屏上某个可以显示的颜色。例如，分辨率为 640×480，每个位平面含有 640×480 位，即有 307 200 位的信息。由于要同时显示 16 种不同颜色，它就具有 4 个彩色位平面，故需要 1 228 800 位的 VRAM，即 153 600B。所以，VRAM 的总容量＝640×480×4b≈150KB。它被分为 4 个位平面，每个位平面提供彩色代码中的一位，每个位平面的容量为 37.5KB。

8.3　典型例题详解

【例 8.1】　试分析图 8-2 所示的写电流波形属于何种记录方式。

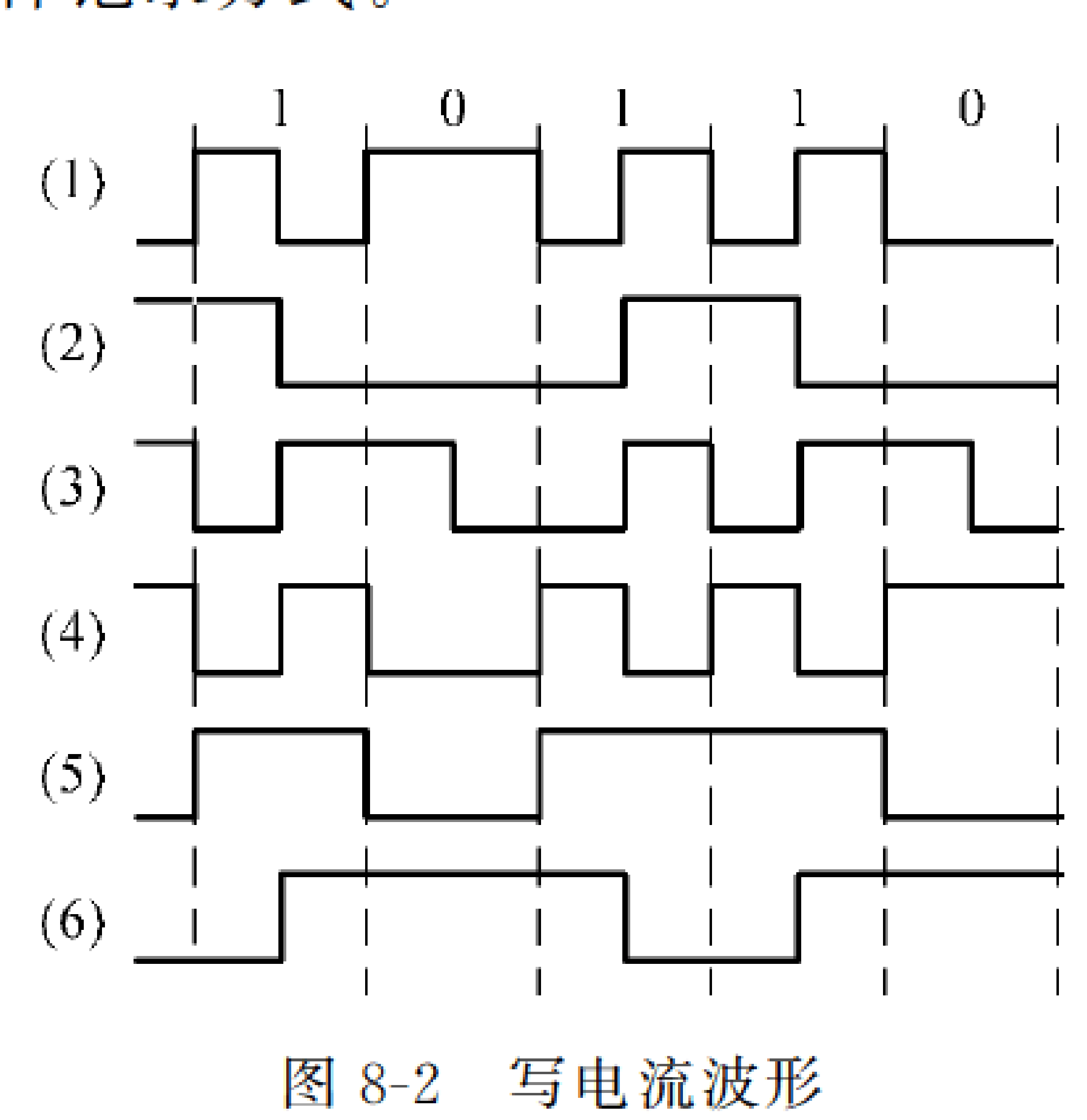


图 8-2　写电流波形

解：(1) 调频制(FM)。记录“1”时，写电流在位周期中间和边界各改变一次方向；记录“0”时，写电流仅在位周期边界改变一次方向，所以记录“1”的磁通翻转频率为记录“0”时的两倍。

(2) 改进调频制(MFM)。记录“1”时，写电流在位周期中间改变方向；记录独立的一个“0”时，写电流不改变方向；记录连续的两个“0”时，写电流在位周期边界改变方向。MFM 制可以减少 FM 制的磁通翻转数目，使之在相同数量的磁通翻转上存储两倍的数据。

(3) 调相制(PE)。记录“1”时，写电流在位周期中间由负变正；记录“0”时，写电流在位周期中间由正

变负。

(4) 调频制(FM)。此波形与波形(1)翻转频率相同,仅相位相反。

(5) 不归零制(NRZ)。记录"1"时,写磁头线圈中通以正向电流;记录"0"时,通以反向电流。

(6) 不归零—1制(NRZ-1)。记录"1"时,在位周期中间写电流改变方向;而记录"0"时,写电流方向维持不变,所以又称为见"1"就翻的不归零制。

【例 8.2】 试推导磁盘存储器读写一块信息所需要的总时间的公式。

解: 设读写一块信息所需要的总时间为 T,平均寻道时间为 T_s,平均等待时间为 T_w,读写一块信息的传输时间为 T_m,则

$$T = T_s + T_w + T_m$$

假设磁盘以每秒 r 转速旋转,每条磁道容量为 N 个字,则数据传输率$=rN$ 字/s。

又假设每块的字数为 n,因而,一旦读写头定位在该块始端就能在 $T_m \approx \left(\frac{n}{rN}\right)$ 的时间内传输完毕。

T_w 是磁盘旋转半圈的时间,$T_w=\left(\frac{1}{2r}\right)$s。由此可得:

$$T = T_s + \frac{1}{2r} + \frac{n}{rN}$$

【例 8.3】 设某磁盘有两个记录面,存储区内直径为 2.36in,外直径为 5in,道密度为 1250TPI,位密度为 52 400bpi,转速为 2400RPM。请解答:

(1) 每面有多少个磁道,每磁道能存储多少字节?

(2) 数据传输率是多少?

(3) 设寻道时间为 10~40ms,在一个磁道上写 8000B 数据,平均需要多少时间?

解: (1) 每面磁道数=道密度×(外直径—内直径)÷2=1250×(5—2.36)÷2=1250×1.32=1650。

通常,位密度是指磁盘最大位密度,即内直径处的位密度。

每道存储的字节数=π×内直径×位密度=π×2.36×52 400b≈48 562B。

(2) 数据传输率=每磁道容量×转速$=48\,562\times\frac{2400}{60}$B/s=1 942 480B/s≈1.94MB/s。

(3) 平均寻道时间$=\frac{t_{smix}+t_{smax}}{2}=\frac{10+40}{2}$ms=25ms。

平均等待时间=磁盘旋转一圈所需时间的一半$=\frac{1}{2\times\frac{2400}{60}}\times 10^3$ms=12.5ms。

数据传输时间$=\frac{\text{传输数据量}}{\text{数据传输率}}=\frac{8000}{1.94\times 10^6}\times 10^3$ms=4.12ms。

写入 8000B 所需时间:

T=平均寻道时间+平均等待时间+数据传输时间=25ms+12.5ms+4.12ms≈41.6ms。

【例 8.4】 某盘组有 5 个盘片,其中有 1 个伺服面,其他盘面为记录数据的盘面,磁盘转速为 7200RPM。盘存储区域内直径为 4.1cm,外直径为 8.9cm,道密度为 40TPM,位密

度为 300bpm，试计算：

(1) 数据盘面数和柱面数是多少？

(2) 盘组容量是多少字节？

(3) 平均等待时间是多少毫秒？

(4) 数据传输率是多少字节/秒？

(5) 给出一个磁盘地址格式方案。

解：(1) 5 个盘片有 10 面，其中伺服面不能存放数据，则共有 9 个数据盘面。

柱面数＝每面上的磁道数＝道密度×(外道半径－内道半径)＝40×(89－41)÷2＝960。

(2) 盘组容量＝数据盘面数×磁道数×内径周长×位密度＝9×960×π×41×300b≈333 694 080b＝41 711 760 B。

(3) 平均等待时间＝旋转一圈时间的一半＝$\frac{60}{2\times7200}$s＝0.004167s＝4.167ms。

(4) 数据传输率为每秒传输的数据量，即每磁道的数据和传输这些数据的时间的比值。

数据传输率＝$\pi\times41\times300\times\frac{7200}{60}$b/s≈4 634 640b/s＝579 330B/s。

(5) 磁盘地址由台号、柱面号、盘面号、扇区号组成。假定只有一台磁盘存储器，所以可以不考虑台号地址。有 10 个记录面，盘面号需 4 位；每个记录面有 960 个磁道，柱面号需 10 位。假定每个扇区记录 512B，则每个磁道有 38 622÷8÷512≈9 个扇区，扇区号需 4 位。由此可得磁盘的地址格式如图 8-3 所示。



图 8-3　磁盘的地址格式

【例 8.5】 某磁盘存储器转速为 3000r/min，共有 4 个记录面，5 道/mm，每道记录信息为 12 288B，最小磁道直径为 230mm，共有 275 道。试问：

(1) 磁盘存储器的容量是多少？

(2) 最高位密度与最低位密度是多少？

(3) 磁盘数据传输率是多少？

(4) 平均等待时间是多少？

(5) 给出一个磁盘地址格式方案。

解：(1) 每道记录信息容量＝12 288B，每个记录面信息容量＝275×12 288B，共有 4 个记录面，所以磁盘存储器的容量＝4×275×12 288 B＝13 516 800B。

(2) 最高位密度 D_1，即最内圈磁道的位密度。

D_1＝每道信息量÷内圈圆周长＝12 288÷(π×最小磁道直径)≈17(B/mm)。

最低位密度 D_2，即最外圈磁道的位密度。

最大磁道半径＝最小磁道半径＋(275÷5)＝(115＋55)mm＝170mm。

D_2＝每道信息量÷外圈圆周长＝12 288÷(π×最大磁道直径)≈11.5B/mm。

(3) 磁盘数据传输率 C＝转速×每道信息容量。

转速 r＝3000÷60＝50r/s，每道信息容量＝12 288B。

$C=50\times12\,288\text{B/s}=614\,400\text{B/s}$。

(4) 平均等待时间$=\frac{1}{2r}=\frac{1}{2\times50}\text{s}=10\text{ms}$。

(5) 假定只有一台磁盘存储器,所以不考虑台号地址。有 4 个记录面,每个记录面有 275 个磁道。假定每个扇区记录 1024B,则需要 12 288÷1024=12 个扇区。由此可以得到的地址格式为:

柱面(磁道)号 9 位,盘面(磁头)号 2 位,扇区号 4 位。

【例 8.6】 有一台磁盘机,其平均寻道时间为 30ms,平均等待时间为 10ms,数据传输率为 500B/ms,磁盘机中随机存放着 1000 块每块为 3000B 的数据。现欲把一块块数据取走,更新后再放回原地。假设一次取出或写入所需时间为:平均寻道时间+平均等待时间+数据传输时间。另外,使用 CPU 更新信息所需时间为 4ms,并且更新时间同输入输出操作不相重叠。试问:

(1) 更新磁盘上的全部数据需多少时间?

(2) 若磁盘机旋转速度和数据传输率都提高一倍,更新全部数据需要多少时间?

解:(1) 由于数据块是随机存放的,所以每取出或写入一块均要定位。

数据传输时间=3000B÷500B/ms=6ms。

更新全部数据所需时间=2×1000×(平均寻道时间+平均等待时间+数据传输时间)+1000×CPU 更新信息时间=[2×1000×(30+10+6)+1000×4]ms=96 000ms=96s。

(2) 磁盘机旋转速度提高一倍后,平均等待时间为 5ms。

数据传输率提高一倍为 1000B/ms,数据传输时间变为 3000÷1000B/ms=3ms。

更新全部数据所需时间=[2×1000×(30+5+3)+1000×4]ms=80 000ms=80s。

【例 8.7】 软盘驱动器使用双面双密度软盘,每面有 80 道,每道 15 个扇区,每个扇区存储 512B。已知磁盘转速 360r/min,假设寻道时间为 10~40ms,今在一个磁道上连续写入 4096B,平均需要多少时间?最长时间是多少?

解:每道存储容量=15×512B=7680B。

磁盘转速=360r/min=6r/s。

数据传输率=7680B×6/s=46 080B/s

读出或写入一块数据所需时间$=\frac{512}{46080}\times1000\text{ms}\approx11.1\text{ms}$。

平均等待时间$=\frac{1}{2\times6}\times1000\text{ms}\approx83.3\text{ms}$。

平均寻道时间$=\frac{10+40}{2}\text{ms}=25\text{ms}$。

4096B 有 4096B÷512B=8 个数据块,根据题意,这 8 个数据块在同一磁道,只需一次定位时间,所以写入 4096B 所需时间=83.3ms+25ms+8×11.1ms≈197ms。

最大等待时间$=\frac{1}{6}\times1000\text{ms}\approx166.6\text{ms}$。

最大寻道时间=40ms。

4096B 有 8 个数据块,写入 4096B 所需最长时间为 166.6ms+40ms+8×11.1ms≈296ms。

【例 8.8】 某打印机定义 94 种 ASCII 码,每行可打印 80 个字符;每个字符采用 7×8 点阵,即横向 7 点,纵向 8 点,问:

(1) 缓存容量有多大?

(2) 字库(ROM)容量有多大?字符"1"(ASCII 码为 31H)在字库中的地址范围为多少?

(3) 缓存中存放的是 ASCII 代码还是点阵信息?缓存地址与打印位置如何对应?

解:(1) 因为每行可打印 80 个字符,所以缓存容量为 80B。

(2) 因为每个字符采用 7×8 点阵,每个字符占 7 个单元,每个单元 8 位,所以列计数器为 3 位,ROM 容量为 94×7×8b=658B。

字符的点阵为 7×8,共需要 7 个地址来存放一个字符的列点阵码。因为"1"的 ASCII 码为 31H,31H-20H=11H,则字符"1"在字库中的地址为 0010001 000~0010001 110,即 088H~08EH。

(3) 缓存中存放的是待打印字符的 ASCII 代码,打印位置自左至右,相应的缓存地址由低到高,每个地址码对应一个字符打印位置。

【例 8.9】 比较光栅扫描的图形显示器与光栅扫描的字符显示器的主要异同点。

解:相同点:按构成一帧显示内容的像素点逐行扫描,显示一帧内容。

不同点:图形显示器需将每个像素的信息都存放在 VRAM 中,而字符显示器只需将要显示的 ASCII 码存放在 VRAM 中,字符的点阵来自字符发生器的 ROM。

【例 8.10】 某字符显示器采用 312 线光栅,每帧画面为 32 字×16 排,上、下各 5 排不显示字符。每个字符按 5×7 点阵组成,每排 12 行光栅,其中 7 行显示字符,5 行为排间间隔。每个字符显示为 6 个点脉冲宽度,其中 5 个点为字符点阵的一行(5 个光点),1 个点为字符间的间隔。设左右边消隐区占 4 个字符时间,水平回扫占 8 个字符时间,垂直回扫占 3 排时间。若主脉冲频率为 4.118MHz,试求:点计数器、字计数器、行计数器、排计数器的计数频率及行同步信号、场同步信号、字符发生器的行选信号的频率。

解:因主脉冲频率为 4.118MHz,字符为 6 个点脉冲,所以点计数器为 6 分频,其频率为 4.118MHz÷6≈0.6863MHz。

行光栅的字符数为 32+4+8=44,所以字计数器为 44 分频,其频率为 0.6863MHz÷44≈15.6kHz。行同步信号频率与此相同。

因每排 12 行光栅,故行计数器为 12 分频,其频率为 15.6kHz÷12≈1.3kHz。字符发生器的行选信号频率与此相同。

整个屏幕的总排数为 16+5+5+3=29,所以排计数器为 29 分频,其频率为 1.3kHz÷29≈44.8Hz。场同步信号频率与此相同。

【例 8.11】 某光栅扫描显示器的分辨率为 1024×1024,帧频为 75Hz(逐行扫描),颜色为 24 位真彩色。回扫和消隐时间忽略不计。试问:

(1) 每一像素允许的读出时间是多少?

(2) 刷新存储器的容量是多少?

(3) 刷新带宽是多少?

(4) 显示总带宽是多少?

解：(1) 每一像素允许的读出时间为 $\frac{1}{75}\times\frac{1}{1024\times1024}s\approx1.27\times10^{-8}s=12.7ns$。

(2) 刷新存储器的容量＝分辨率×颜色深度＝1024×1024×24b＝1024×1024×3B＝3MB。

(3) 刷新带宽＝分辨率×颜色深度×帧频＝1024×1024×3B×75/s＝235 929 600B/s＝225MB/s。

(4) 显示总带宽＝刷新带宽＝225MB/s。

【例 8.12】 某一汉字 CRT 显示器(字符方式显示)，可显示 3000 个汉字，每字以 11×16 点阵组成，字间间隔一点，两排字间隔 4 线，32 字/排，12 排/屏。一个汉字编码占 2 个字节。帧频 50Hz。帧回扫和行回扫均占扫描时间的 20%(扫描时间包括正扫和回扫)。试求：

(1) VRAM 的容量是多少？

(2) 字符发生器(ROM)的容量是多少？

(3) 各计数器的位数分别是多少？时钟频率是多少？

解：(1) VRAM 存储器存储汉字的编码，因为一屏可显示 32×12＝384 字，每个汉字的编码占 2 个字节，所以 RAM 的容量＝32×12×2B＝768B。

(2) ROM 存储器存储汉字的行点阵信息，因为总共可显示 3000 个汉字，每个汉字以 11×16 点阵组成，所以 ROM 的容量＝3000×11×16b。

(3) 排计数器：汉字可显示 12 排，根据帧回扫占扫描时间 20%，可求得回扫占 3 排时间，所以一共是 15 排，则排计数器需要 4 位。

行计数器：每个汉字点阵 16 行，两排字间隔 4 行，故一排汉字共用 20 行，行计数器需要 5 位。

字计数器：每排 32 个字，根据行回扫占扫描时间 20%，可求得回扫为 8 个字时间，故共 40 个字，则字计数器需用 6 位。

点计数器：每个字的点阵是 11 列，加上间隔 1 点，共 12 点，故点计数器为 4 位。

这样时钟频率为 15×20×40×12×50Hz＝7 200 000Hz＝7.2MHz。

【例 8.13】 一级汉字有 3755 个，如每个汉字字模采用 16×16 点阵，并存放在主存中，问约占多少字节？假设将汉字显示在荧光屏上，共 24 行，每行 80 个字，为保存一帧信息，约需多少字节的存储空间(不考虑颜色数)？

解：汉字字模的容量＝3755×32≈120KB。

显示存储器的容量＝24×80×32＝60KB(图形显示器)。

显示存储器的容量＝24×80×2≈3.8KB(字符显示器)。

*【例 8.14】 假定一台计算机的显示存储器用 DRAM 芯片实现，若要求显示分辨率为 1600×1200，颜色深度为 24 位，帧频为 85Hz，显存总带宽的 50%用来刷新屏幕，则需要的显存总带宽至少约为________。

A. 245Mbps　　B. 979Mbps　　C. 1958Mbps　　D. 7834Mbps

解：D。

分析：显存带宽＝分辨率×色深×帧频，考虑到 50%的时间用来刷新屏幕，故显存总带宽应当加倍。所以有显存带宽＝1600×1200×24b×85Hz＝3916.8Mbps，则需要的显存总

带宽为 3916.8÷0.5=7833.6Mbps ≈ 7834Mbps。注意，题干中 4 个选项的单位均为兆位每秒，而不是兆字节每秒。

*【例 8.15】 下列选项中，用于提高 RAID 可靠性的措施有________。

Ⅰ. 磁盘镜像　Ⅱ. 条带化　Ⅲ. 奇偶检验　Ⅳ. 增加 Cache 机制

A. 仅Ⅰ、Ⅱ　B. 仅Ⅰ、Ⅲ　C. 仅Ⅰ、Ⅲ和Ⅳ　D. 仅Ⅱ、Ⅲ和Ⅳ

解：B。

分析：提高 RAID 可靠性的措施有磁盘镜像和奇偶校验。其中，磁盘镜像具有最高的可靠性，但只有一半的磁盘空间被用来存储数据。主要用在对数据安全性要求很高，而且要求能够快速恢复被损坏数据的场合。

*【例 8.16】 某磁盘的转速为 10 000r/min，平均寻道时间是 6ms，磁盘传输速率是 20MB/s，磁盘控制器延迟为 0.2ms，读取一个 4KB 的扇区所需的平均时间约为________。

A. 9ms　B. 9.4ms　C. 12ms　D. 12.4ms

解：B。

分析：读取一个扇区的平均时间应该包括平均寻道时间、平均等待时间、数据传输时间和磁盘控制器延迟 4 个部分。其中，平均等待时间是磁盘旋转半圈的时间，数据传输时间等于传送的数据量除以磁盘传输速率。因为磁盘转速为 10000r/min=166.67r/s，故旋转半圈的时间为 3ms。数据传输时间=4KB÷20MB/s=0.2ms。所以，读取一个扇区的平均时间=9ms+3ms+0.2ms+0.2ms=9.4ms。

8.4 同步测试习题及解答

8.4.1 同步测试习题

一、填空题

1. 在磁介质存储器中，记录方式可以分为多个大类，每类中又衍生出若干派生方案。其中，调频制(FM)记录方式目前主要用于单密度磁盘存储器，________记录方式主要用于双密度磁盘存储器，而在磁带存储器中一般采用________记录方式。

2. 在磁介质存储器中，格式化容量是指________。

3. 对于文本模式的打印机，主机送往打印机的应当是打印字符的________码。

4. 在字符打印机或显示器的字库中，存放着字符的________。

5. 字符显示器中的 VRAM 用来存放________。

二、选择题

1. 计算机的外围设备是指________。

A. 输入输出设备　B. 外存储器

C. 输入输出设备和外存储器　D. 电源

2. 在输入输出设备中，________是复合型的输入输出设备。

A. 鼠标　B. 磁盘

C. 打印机　D. CD-ROM 光盘

3. 带有处理器的终端一般被称为________。

A. 交互式终端　B. 智能终端　C. 远程终端　D. 联机终端

4. 在调频制记录方式中，是利用________来写 0 或 1。

A. 电平高低的变化　B. 电流幅值的变化

C. 电流相位的变化　D. 电流频率的变化

5. 下列各种记录方式中，不具备自同步能力的是________。

A. 不归零制 NRZ　B. 改进型调频制 MFM

C. 调相制 PM　D. 调频制 FM

6. 磁盘存储器的平均等待时间通常是指________。

A. 磁盘旋转一周所需的时间　B. 磁盘旋转半周所需的时间

C. 磁盘旋转 1/3 周所需的时间　D. 磁盘旋转 2/3 周所需的时间

7. 活动头磁盘存储器的寻道时间通常是指________。

A. 最小寻道时间

B. 最大寻道时间

C. 最大寻道时间和最小寻道时间的平均值

D. 最大寻道时间和最小寻道时间之和

8. 若磁盘的转速提高一倍，则________。

A. 平均等待时间减半　B. 存储密度提高一倍

C. 平均寻道时间减半　D. 磁盘访问速度提高一倍

9. 磁盘的盘面上有很多半径不同的同心圆组成，这些同心圆称为________。

A. 扇区　B. 磁道　C. 柱面　D. 磁表面

10. 在下列存储器中，若按存取速度从快到慢的顺序排列，应当为________。

A. 高速缓存、寄存器组、主存、磁带、软磁盘、活动头硬磁盘

B. 寄存器组、高速缓存、主存、磁带、软磁盘、活动头硬磁盘

C. 寄存器组、高速缓存、主存、软磁盘、活动头硬磁盘、磁带

D. 寄存器组、高速缓存、主存、活动头硬磁盘、软磁盘、磁带

11. 为提高存储器存取效率，在安排磁盘上信息分布时，通常是________。

A. 存满一面，再存另一面

B. 尽量将同一文件存放在一个扇区或相邻扇区的各磁道上

C. 尽量将同一文件存放在不同面的同一磁道上

D. 上述方法均有效

12. 某磁盘的转速为 7200r/min，传输速度为 4MB/s，控制器开销为 1ms。要保证读或写一个 512B 的扇区的平均时间为 11.3ms。那么，该磁盘的平均寻道时间不超过________。

A. 3.9 ms　B. 4.7 ms　C. 5.5 ms　D. 6.1 ms

13. PC 键盘常常采用单片机作为键盘控制器，它通过一条 5 芯电缆向主机提供闭合键的________。

A. 二进制代码　B. BCD 码　C. ASCII 码　D. 扫描码

14. 对于字符显示器，主机送给显示器的应是显示字符的________。

A. ASCII 码　B. 列点阵码　C. BCD 码　D. 行点阵码

15. CRT 图形显示器的分辨率表示为________。

A. 一个图像点(像素)的物理尺寸

B. 显示器一行能显示的最大图像点数与一列能显示的最大图像点数

C. 显示器屏幕可视区域的大小

D. 显示器能显示的字符个数

16. 显示器的灰度级是指________。

A. 显示器的亮度　　B. 显示字符清晰程度

C. 显示器中光点亮暗的层次级别　　D. 显示器中显示存储器的容量

17. 显示汉字也是采用汉字点阵原理,若每个汉字用16×16的点阵表示,7500个汉字的字库容量是________。

A. 16KB　　B. 240KB　　C. 320KB　　D. 1MB

18. 一台显示器的图形分辨率为1024×768,要求显示256种颜色,显示存储器VRAM的容量至少为________。

A. 512KB　　B. 1MB　　C. 3MB　　D. 4MB

三、判断题

1. 改进的调频制是目前软盘中采用的磁记录方式。　(　　)

2. 通常磁盘存储器每条磁道的存储容量是相同的。　(　　)

3. 磁带和磁盘都是直接存储设备。　(　　)

4. $M \times N$ 点阵针式串行打印机的打印头上,装有 M 根打印针。　(　　)

5. 打印机字库中存放着字形的行点阵码。　(　　)

6. 激光打印机是高速的击打式打印机。　(　　)

7. 字符显示器的字库中存放着字形的列点阵信息。　(　　)

8. 利用光学方式读写信息的存储器称为光盘。　(　　)

四、简答题

1. 简述分辨率、灰度级的概念以及它们对显示器性能的影响。

2. 为什么要不断地对CRT屏幕进行刷新?要求的刷新频率是多少?为达此目的,必须设置什么样的硬件条件?

五、综合题

1. 图8-4所示为同一数据的几种写电流波形。试判断它们各是什么磁记录方式,并且指明有无自同步能力。

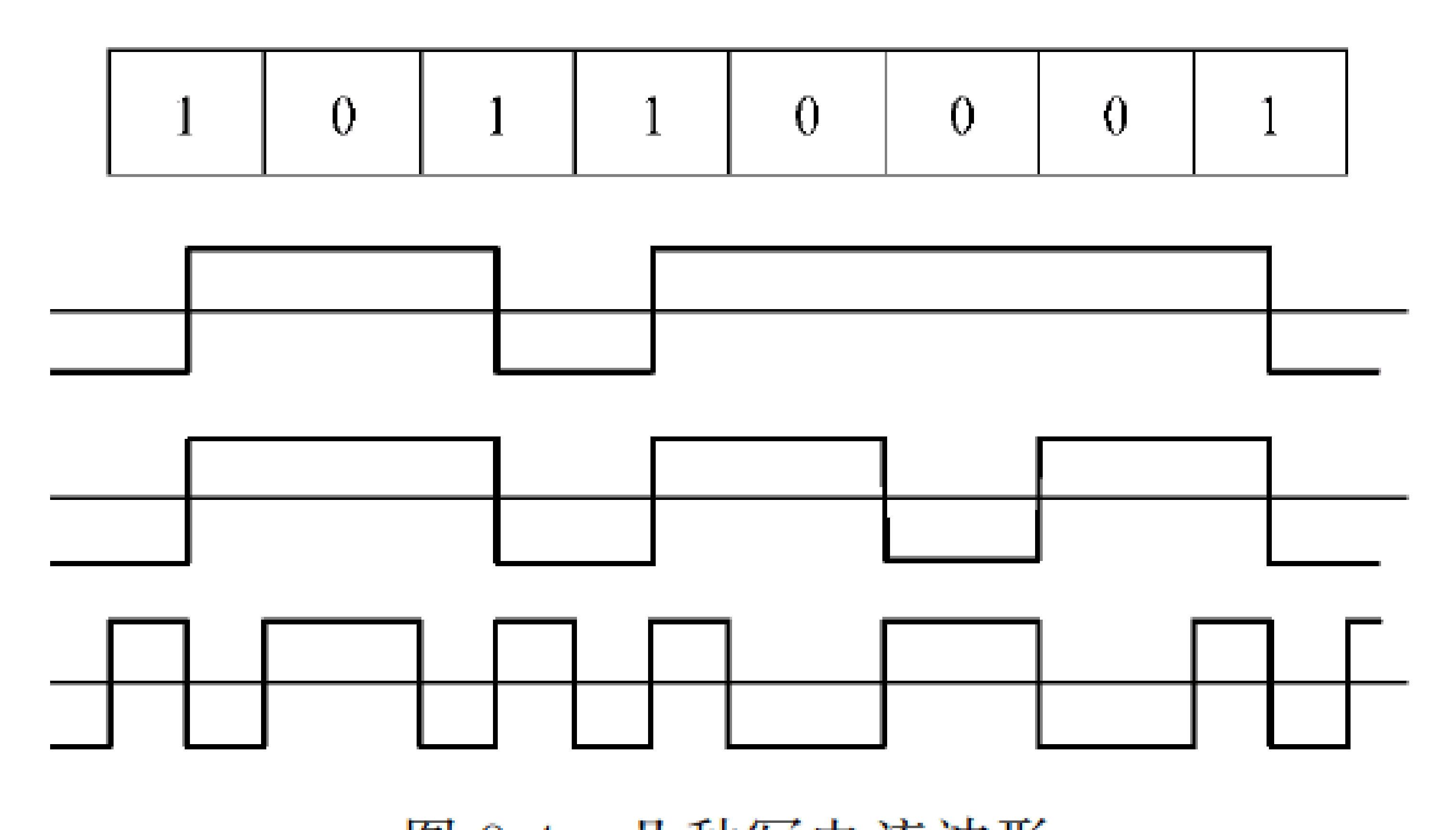


图8-4　几种写电流波形

2. 某磁盘存储器的转速为 nr/min,共有4个记录盘面,每道记录信息为 mB,共256道,

试问：

（1）磁盘存储器的存储容量是多少？

（2）磁盘数据传输率是多少？

（3）平均等待时间是多少？

3．一台有3个盘片的磁盘组，共有4个记录面，转速为7200r/min，盘面有效记录区域的外直径为30cm，内直径为20cm，记录位密度为250b/mm，磁道密度为8道/mm，盘面分16个扇区，每扇区1024B，设磁头移动速度为2m/s。

（1）试计算盘组的非格式化容量和格式化容量。

（2）计算该磁盘的数据传输率、平均寻道时间和平均等待时间。

（3）若一个文件超出1个磁道的容量，余下的部分是存放于同一盘面上还是存放于同一柱面上？请给出一个合理的磁盘地址方案。

4．某活动磁头磁盘组的性能参数为转速6000r/min，每道容量20KB，平均寻道时间10ms。假定操作系统采用两种方式访问该磁盘组，第一种方式每次读取2KB，第二种方式每次读取8KB。已知操作系统访问磁盘的时间开销为每次5ms的额外开销和每2KB 1ms的传输开销。

（1）计算该磁盘组的数据传输率和平均等待时间是多少（即从磁头定位到目标磁道开始到寻找到目标扇区的平均时间）？

（2）如果要读取一个很大的磁盘文件，操作系统按第一种方式和第二种方式所需要的时间比是多少？

5．如果某计算机显示器的分辨率为1024×768，65536灰度级，则它显示卡的刷新存储器至少为多少容量？

6．一个黑白CRT，显示具有16级灰度的图片，已知CRT的分辨率为800×600，问显示RAM的容量为多少？如帧同步脉冲的频率为30Hz，则视频脉冲的频率应是多少？

7．若需显示一幅1024×768像素有256种颜色的图像。试问：

（1）显示系统的帧存容量为多少位？

（2）如果要在屏幕上得到逼真的动态图像，假设传送50帧/s（逐行扫描），则传送速率为多少？

（3）如果要显示汉字，机器内设置有ROM汉字库，存放一级和二级汉字，汉字采用16×16点阵，则汉字库的容量是多少？

8．在GB2312—80标准中定义了一级汉字3755个，二级汉字3008个，若每个汉字字模采用16×16点阵，则两级汉字各占多少存储器空间？若将汉字显示在屏幕上，共24行，每行80个字，为保存一屏信息，共需要多大的存储空间？

8.4.2 同步测试习题解答

一、填空题

1．改进的调频制（MFM），调相制（PE）。

2．用户实际可以使用的存储容量。

3．ASCII码。对于文本（字符）模式的打印机，主机送出的是打印字符的ASCII码，然后由打印机中的字库将其转换成相应字符的字模点阵码。

4. 字模点阵码。通常打印字库中是列点阵码,显示字库中是行点阵码。

5. 显示字符的 ASCII 码。在字符显示器中,主机送出的显示字符的 ASCII 码被存放在显示器的 VRAM 中,为不断地刷新画面提供信息,以保证显示稳定的画面。

二、选择题

1. C。除主机以外,围绕着主机设置的各种硬件装置均称为外围设备。

2. B。磁盘既可以读(输入)又可以写(输出),所以是复合型的输入输出设备。

3. B。智能终端中一定带有处理器。

4. D。在调频制记录方式中,信息的写入是依靠写入电流频率的变化来实现的,写1时的电流变化频率是写0时电流变化频率的2倍。

5. A。自同步能力是指从读出的数据中自动提取同步信号的能力。在各种记录方式中,NRZ,NRZ-1 记录方式没有自同步能力,PM、FM、MFM 记录方式具有自同步能力。

6. B。磁盘的平均等待时间是最小等待时间(不旋转)与最大等待时间(旋转一圈时间)相加之后除以2。

7. C。磁盘的平均寻道时间是最小寻道时间与最大寻道时间相加之后除以2。

8. A。磁盘的转速提高可以减少平均等待时间。

9. B。磁盘的盘面上的同心圆就是磁道。

10. D。存取速度由快至慢依次为寄存器组、高速缓存、主存、活动头硬磁盘、软磁盘、磁带。

11. C。如果选择同一圆柱面上的不同磁道,由于各记录面的磁头已同时定位,换道的时间只是磁头选择电路的译码时间,相对于定位操作可以忽略不计。

12. D。磁盘的平均存取时间=平均寻道时间+平均等待时间+控制器开销+读写时间。其中,平均等待时间=[60÷(2×7200)]ms≈4.17ms,读写时间=512B÷4MB/s≈0.122ms。

平均寻道时间=磁盘的平均存取时间-平均等待时间-控制器开销-读写时间=(11.3-4.17-1-0.122)ms=6.008ms。

13. D。键盘向主机接口电路提供的是串行扫描码。

14. A。主机送给字符显示器的是显示字符的 ASCII 码。

15. B。图形显示器的分辨率以水平显示的像素个数×水平扫描线数表示。

16. C。像素具有明暗和色彩属性。明暗变化的数量称为灰度级,对于彩色显示器就是颜色数。

17. B。每个16×16点阵汉字需32B,汉字的字库容量=7500×32B=240 000B。

18. B。显示存储器(VRAM)容量为 $1024\times768\times\log_2 256\text{b}=1024\times768\times8\text{b}=768\text{KB}\approx1\text{MB}$。

三、判断题

1. √。

2. √。

3. ×。磁盘是直接存储设备,磁带是顺序存储设备。

4. ×。$M\times N$ 点阵针式串行打印机的打印头上,装有 N 根打印针。

5. ×。打印机字库中存放着字形的列点阵码。

6. ×。激光打印机是高速的非击打式打印机。

7. ×。字符显示器的字库中存放着字形的行点阵信息。

8. √。

四、简答题

1. 分辨率是衡量显示器显示清晰度的指标，以像素的个数为标志。显示器中显示的像素越多，分辨率就越高，显示的文字和图像就越清晰。灰度级是指显示器所显示的像素点的亮度差别。显示器的灰度级越高，显示的图像层次就越丰富逼真。

2. CRT 显示器发光是由于电子束打在荧光粉上引起的。电子束扫过之后其发光亮度只能维持几十 ms。为了使人眼能看到稳定的显示图像，必须使电子束不断地重复扫描整个屏幕，这个过程称为刷新。通常，刷新频率要求大于 50 帧/s 图像。刷新频率越高，屏幕的闪烁越小，对人眼睛产生的刺激越小。早期显示器通常支持 50～60Hz 的刷新频率，现在 VESA（视频电子标准协会）规定 85Hz 为无闪烁的刷新频率。为了不断提高刷新图像的信号，必须把一帧图像的信息存储在刷新存储器（VRAM）中。

五、综合题

1. 图 8-4 所示的几种写电流波形分别为不归零—1 制、改进的调频制和调频制。不归零—1 制没有自同步能力，改进的调频制和调频制有自同步能力，但改进的调频制的自同步能力低于调频制。

2. (1) 存储容量 $=m\times256\times4\text{B}=1024m\text{B}$。

(2) 磁盘数据传输率 $=m\times\dfrac{n}{60}\text{B/s}=\dfrac{mn}{60}\text{B/s}$。

(3) 平均等待时间 $=\dfrac{1}{2}\times\dfrac{60}{n}\text{s}=\dfrac{30}{n}\text{s}$

3. (1) 磁盘的记录区域＝(30－20)cm÷2＝5cm。

磁盘的磁道数＝50mm×8 道/mm＝400 道。

每道的非格式化容量＝π×200mm×250b/mm≈157 000b＝19 625B。

每道格式化容量＝16×1024B＝16 384B。

盘组的非格式化容量＝4×400×19 625B＝31 400 000B≈30MB。

盘组的格式化容量＝4×400×16 384B＝26 214 400B＝25MB。

(2) 磁盘的数据传输率＝16 384B×7200r/min＝1 966 080B/s＝1.875MB/s。

因为磁头移动 400 道的时间＝50mm÷2000mm/ms＝25ms，

平均寻道时间 $=\dfrac{25}{2}\text{ms}=12.5\text{ms}$。

因为磁盘旋转一周的时间为 60s÷7200≈8.3ms，

平均等待时间＝8.3ms÷2≈4.15ms。

(3) 若一个文件超出 1 个磁道的容量，余下的部分存于同一柱面上。磁盘地址格式为磁道号 9 位（选 400 个磁道），盘面号 2 位（选 4 个记录面），扇区号 4 位（选 16 个扇区）。

4. (1) 数据传输率 $=\dfrac{20\times1024}{\dfrac{60}{6000}}\text{MB/s}\approx2\text{MB/s}$。

平均等待时间 $=\dfrac{60}{2\times6000}\text{s}=5\text{ms}$。

(2) 操作系统按第一种方式每读取2KB的时间=平均寻道时间+平均等待时间+2KB的传输时间+额外开销时间=(10+5+1+5)ms=21ms。

操作系统按第二种方式每读取8KB的时间=平均寻道时间+平均等待时间+8KB的传输时间+额外开销时间=(10+5+4+5)ms=24ms。

设读取的磁盘文件长度为NKB,则第一种方式可分为$\frac{N}{2}$次传送,第二种方式可分为$\frac{N}{8}$传送。

操作系统按第一种方式所需要的时间=$\frac{N}{2}\times 21$,操作系统按第二种方式所需要的时间=$\frac{N}{8}\times 24$,则操作系统按第一种方式和第二种方式所需要的时间比为3.5。

5. 刷新存储器(VRAM)容量为$(1024\times 768\times \log_2 65536)b=1024\times 768\times 16$b=1536KB。

6. 显示RAM容量为$(800\times 600\times \log_2 16)b=800\times 600\times 4$b≈234KB。

视频脉冲的频率为$(800\times 600\times 30)$Hz=14.4MHz。

7. (1) 显示RAM的容量为$(1024\times 768\times \log_2 256)b=1024\times 768\times 8$b=768KB。

(2) 传送速率为50×768KB/s=37.5MB/s。

(3) 因为一级汉字个数为3755个,二级汉字为3008个,所以汉字库的容量为$(3755+3008)\times 16\times 16$b≈211KB。

8. 一级字库120160B,二级字库96256B,每屏显示汉字1920个,每个汉字仅需要保存2个字节的内码,显示存储器的存储空间为3840B。

注意:*在存储容量和数据传输率的计算中*,1K=1024。

第 9 章 输入输出系统

9.1 基本内容摘要

- 主机与外设的连接
  - 输入输出接口
  - 接口的功能和基本组成
  - 外设的识别与端口寻址
  - 输入输出信息传送控制方式
- 程序查询方式及其接口
  - 程序查询方式
  - 程序查询方式接口
- 中断系统和程序中断方式
  - 中断的基本概念
  - 中断请求和中断判优
  - 中断响应和中断处理
  - 多重中断与中断屏蔽
  - 中断全过程
- DMA 方式及其接口
  - DMA 方式的基本概念
  - DMA 接口
  - DMA 传送方法与传送过程
- 通道控制方式
  - 通道的基本概念
  - 通道的类型与结构
  - 通道程序
  - 通道工作过程

9.2 重点难点梳理

1. 接口与端口

接口(interface)与端口(port)是两个不同的概念。输入输出接口(I/O 接口)是主机和外设之间的交接界面,通过接口可以实现主机和外设之间的信息交换。端口是指接口电路中可以被 CPU 直接访问的寄存器,若干个端口加上相应的控制逻辑电路才组成接口。

通常,一个接口中包含数据端口、命令端口和状态端口。存放数据信息的寄存器称为数据端口,存放控制命令的寄存器称为命令端口,存放状态信息的寄存器称为状态端口。CPU 通过输入指令可以从有关端口中读取信息,通过输出指令可以把信息写入有关端口。CPU 对不同端口的操作有所不同,有的端口只能写或只能读,有的端口既可以读又可以写。例如,对状态端口只能读,可将外设的状态标志送到 CPU 中;对命令端口只能写,可将 CPU 的各种控制命令发送给外设。为了节省硬件,在有的接口电路中,状态信息和控制信息可以共用一个寄存器(端口),称为设备的控制/状态寄存器。

2. 独立编址方式的端口访问

独立编址方式在 Intel 系列、Z80 系列微机以及大型计算机中得到广泛应用。例如,Intel 80x86 的 I/O 地址空间由 2^{16}(64K)个独立编址的 8 位端口组成。两个连续的 8 位端口可作为 16 位端口处理;4 个连续的 8 位端口可作为 32 位端口处理。因此,I/O 地址空间最多能提供 64K 个 8 位端口、32K 个 16 位端口、16K 个 32 位端口或总容量不超过 64KB 的不同端口的组合。

80x86 的专用 I/O 指令 IN 和 OUT 有直接寻址和间接寻址两种类型。直接寻址 I/O 端口的寻址范围为 00～FFH,至多为 256 个端口地址。这时程序可以指定:

编号为 0～255 的 256 个 8 位端口;

编号为 0、2、4、…、252、254 的 128 个 16 位端口;

编号为 0、4、8、…、248、252 的 64 个 32 位端口。

间接寻址由 DX 寄存器间接给出 I/O 端口地址。DX 寄存器长 16 位,所以间接寻址 I/O 端口的寻址范围为 0000～FFFFH,最多为 2^{16}=64K 个端口地址。这时程序可指定:

编号为 0～65535 的 65536 个 8 位端口;

编号为 0、2、4、…、65532、65534 的 32768 个 16 位端口;

编号为 0、4、8、…、65528、65532 的 16384 个 32 位端口。

CPU 一次可实现字节(8 位)、字(16 位)或双字(32 位)的数据传送。与存储器中的双字一样,32 位端口应对准可被 4 整除的偶地址;与存储器中的字一样,16 位端口应对准偶地址;8 位端口可定位在偶地址,也可定位在奇地址。

3. 程序查询方式的工作过程

1) 预置传送参数

在传送数据之前,由 CPU 执行一段初始化程序,预置传送参数。传送参数包括存取数据的主存缓冲区首地址和传送数据的个数。

2) 向外设接口发出命令字

当 CPU 选中某台外设时,执行输出指令向外设接口发出命令字启动外设,为接收数据

或发送数据做应有的操作准备。此项任务由输出指令完成，将命令字送至接口的命令端口。

3）从外设接口取回状态字

CPU 执行输入指令，从外设接口中取回状态字并进行测试，判断数据传送是否可以进行。此项任务由输入指令完成，将接口的状态端口中的状态字送至 CPU。

4）查询外设标志

CPU 不断查询状态标志。如果外设没有准备就绪，CPU 就踏步进行等待，一直到这个外设准备就绪，并发出“外设准备就绪”信号为止。

5）传送数据

只有外设准备好，才能实现主机与外设之间的一次数据传送。输入时，CPU 执行输入指令，从外设接口的数据缓冲寄存器中接收数据；输出时，CPU 执行输出指令，将数据写入外设接口的数据缓冲寄存器中。

6）修改传送参数

每进行一次数据传送之后必须要修改传送参数，其中包括主存缓冲区地址加 1，传送个数计数器减 1。

7）判断传送是否结束

如果传送个数计数器不为 0，则转第 3)步，继续传送，直到传送个数计数器为全 0，表示传送结束。

4. 中断的基本概念

中断是指当计算机执行现行程序时，系统中出现某些急需处理的异常情况和特殊请求，CPU 暂时中止现行程序，而转去对随机发生的更紧迫的事件进行处理，在处理完毕后，CPU 将自动地返回原来的程序继续执行。

中断的类型有以下基本类型。

1）自愿中断和强迫中断

自愿中断又称程序自中断，它不是随机产生的中断，而是在程序中安排软中断指令，这些指令可以使机器进入中断处理的过程。

强迫中断是随机产生的中断，当这种中断产生后，由中断系统强迫计算机中止现行程序并转入中断服务程序。

2）程序中断和简单中断

程序中断是指主机在响应中断请求后，通过执行一段中断服务程序来处理这一任务，它需要占用一定的 CPU 时间。

简单中断就是外设与主存之间直接进行信息交换的方法，即 DMA 方式。这种“中断”不去执行中断服务程序，故不破坏现行程序的状态。主机发现有简单中断请求（也就是 DMA 请求）时，将让出一个或几个存取周期供外设与主存交换信息，然后继续执行程序。

3）内中断和外中断

内中断是指由于 CPU 内部硬件或软件原因引起的中断，例如单步中断、溢出中断等。

外中断是指 CPU 以外的部件引起的中断。通常，外中断又可以分为不可屏蔽中断和可屏蔽中断两种。不可屏蔽中断优先级别较高，常用于应急处理，例如掉电、主存读写校验错等；而可屏蔽中断级别较低，常用于一般 I/O 设备的数据传送。

4）向量中断和非向量中断

向量中断是指那些中断服务程序的入口地址是由中断事件自己提供的中断。中断事件在提出中断请求的同时,通过硬件向主机提供中断服务程序入口地址或指针,即向量地址。

非向量中断的中断事件不能直接提供中断服务程序的入口地址。

5）单重中断和多重中断

单重中断在CPU执行中断服务程序的过程中不能再被打断。

多重中断在执行某个中断服务程序的过程中,CPU可转去响应级别更高的中断请求,又称为中断嵌套。

5. 程序中断方式与调用子程序的区别

计算机的中断处理过程有点类似于调用子程序的过程,这里现行程序相当于主程序,中断服务程序相当于子程序。它们的主要区别在于:

(1) 子程序的执行是由程序员事先安排好的(由一条调用子程序指令转入),而中断服务程序的执行则是由随机的中断事件引起的。

(2) 子程序的执行受到主程序或上层子程序的控制,而中断服务程序一般与被中断的现行程序毫无关系。

(3) 不存在同时调用多个子程序的情况,而有可能发生多个外设同时请求CPU为自己服务的情况。

6. 程序中断方式和程序查询方式的比较

(1) 在程序查询I/O方式中,何时对何设备进行输入和输出操作完全受CPU控制;在程序中断I/O方式中,何时对设备操作由外设主动通知CPU。

(2) 程序查询I/O方式中,CPU与外设不能并行工作;程序中断I/O方式中,CPU与外设可以并行工作。

(3) 程序查询方式无法处理异常事件,例如掉电、非法指令、地址越界等;程序中断方式可以处理这些异常事件。

(4) 程序查询方式的优点是硬件结构比较简单,缺点是CPU效率低而且只能进行数据传送;程序中断方式硬件结构相对复杂一些。

7. CPU响应中断的条件

1）CPU接收到中断请求信号

中断源要发出中断请求,相应的中断请求触发器处于“1”状态。

2）CPU允许中断

CPU允许中断,即开中断。CPU内部有一个中断允许触发器(EINT),只有当EINT=1时,CPU才可以响应中断源的中断请求(中断允许);如EINT=0,CPU处于不允许中断状态,即使中断源有中断请求,CPU也不响应(中断关闭)。

3）一条指令执行完毕

这是CPU响应中断请求的时间限制条件。一般情况下,CPU在一条指令执行完毕且没有更紧迫的任务时才能响应中断请求。

8. 中断隐指令

CPU响应中断之后,经过某些操作,转去执行中断服务程序。这些操作是由硬件直接实现的,一般称为中断隐指令。中断隐指令并不是指令系统中的一条真正的指令,它没有操

作码，所以中断隐指令是一种不允许、也不可能为用户使用的特殊指令。中断隐指令主要完成以下的操作。

1）保存断点

为了保证在中断服务程序执行完毕能正确地返回原来的程序，必须将原来程序的断点（即程序计数器 PC 的内容）保存起来。断点可以压入堆栈，也可以存入主存的特定单元中。

2）暂不允许中断

暂不允许中断即关中断。在中断服务程序中，为了保护中断现场（即 CPU 的主要寄存器状态）期间不被新的中断所打断，必须要关中断，从而保证被中断的程序在中断服务程序执行完毕之后能接着正确地执行下去。

3）引出中断服务程序

引出中断服务程序的实质就是取出中断服务程序的入口地址送程序计数器 PC。

通常，中断隐指令的任务在中断周期内来完成，如图 9-1 所示。

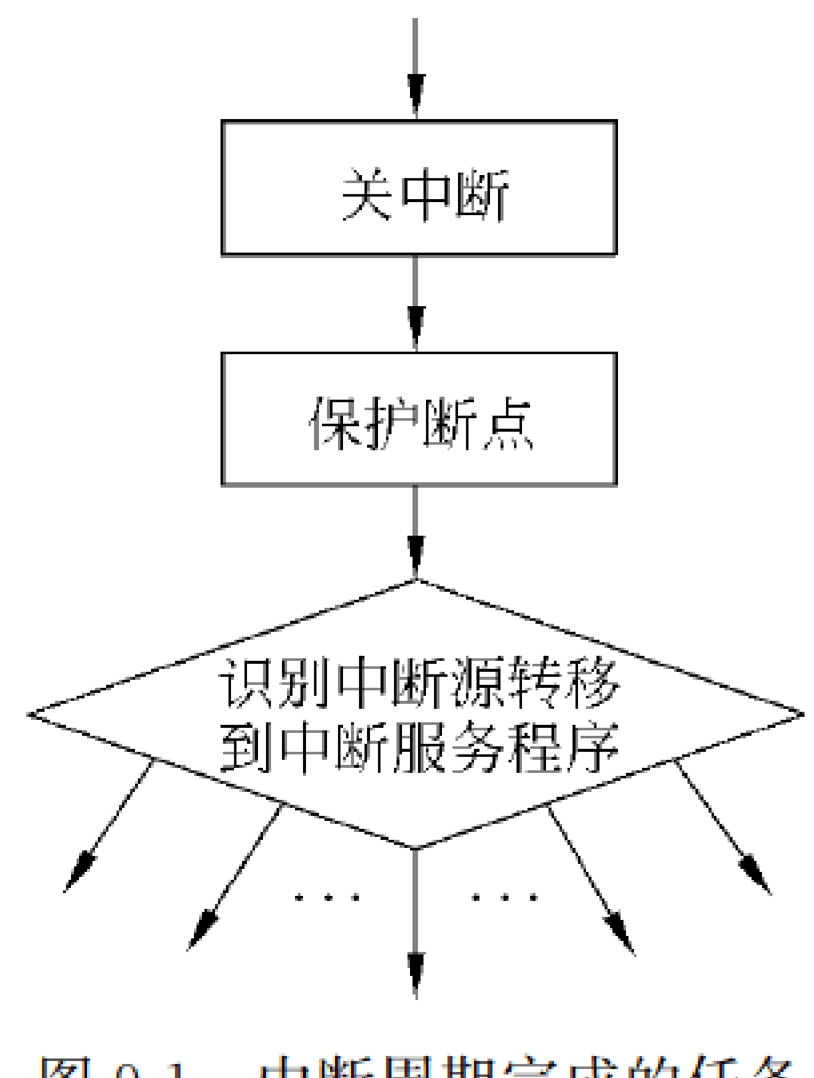


图 9-1　中断周期完成的任务

9. 中断向量地址

向量中断是指那些中断服务程序的入口地址是由中断事件自己提供的中断。中断事件在提出中断请求的同时，通过硬件向主机提供中断服务程序入口地址或指针，即向量地址。

1）向量地址是中断服务程序的入口地址

如果向量地址就是中断服务程序的入口地址，则 CPU 不需要再经过处理就可以进入相应的中断服务程序。

2）向量地址是中断向量表的指针

如果向量地址是中断向量表的指针，则向量地址指向一个中断向量表，从中断向量表的相应单元中再取出中断服务程序的入口地址，此时中断源给出的向量地址是中断服务程序入口地址的地址。

中断向量表通常是在主存中开辟的一块存储空间，用来存放中断服务程序的入口地址信息，如图 9-2 所示。在 CPU 响应中断后，中断硬件自动地将向量地址送到数据总线上，CPU 读数据总线获得向量地址，然后根据向量地址查询向量表获得中断服务程序的入口地址，从而转入中断服务程序。

图 9-2　中断向量表

例如，80x86 的中断向量表占用主存 00000H～003FFH 共 1KB 的存储空间，表中内容分为 256 项，对应于中断类型号 0～255，每一项占 4 个字节，高地址的两个字节用来存放中断服务程序所在段的段地址，低地址的两个字节用来存放中断服务程序入口处所在段的偏移地址。从中断向量表的结构可知，n 号中断服务程序的入口地址存放在表中 $4\times n$～$4\times n+3$共 4 个字节中。

10. 允许和禁止中断

允许中断还是禁止中断是用 CPU 中的中断允许触发器控制的,当中断允许触发器(EINT)被置"1",则允许中断;当中断允许触发器(EINT)被置"0",则禁止中断。

允许中断即开中断,下列情况应开中断:

(1) 无论是单重中断还是多重中断,在中断服务程序执行完毕,恢复中断现场之后。

(2) 在多重中断的情况下,保护中断现场之后。

禁止中断即关中断,下列情况应关中断:

(1) 当响应某一级中断请求,不再允许被其他中断请求打断时。

(2) 在中断服务程序的保护和恢复现场之前。

开中断与关中断的任务通常由专门的指令来完成。

11. 中断请求和中断屏蔽

中断源发出中断请求之后,这个中断请求并不一定能真正送到 CPU 中,在有些情况下,可以用程序方式有选择地封锁部分中断源的中断请求,而对其余中断源的中断请求继续开放,这就是中断屏蔽。

如果给每个中断源都相应地配备一个中断屏蔽触发器(MASK),则每个中断请求信号在送往判优电路之前,还要受到屏蔽触发器的控制。当 MASK=1,表示对应中断源的请求被屏蔽(封锁其中断源的请求)。中断请求触发器和中断屏蔽触发器是成对出现的,只有当 $INTR_i=1$(中断源有中断请求),$MASK_i=0$(该级中断未被屏蔽),才允许对应的中断请求送往 CPU,所以 $INT_i=INTR_i \cdot \overline{MASK_i}$,相应的电路如图 9-3 所示。

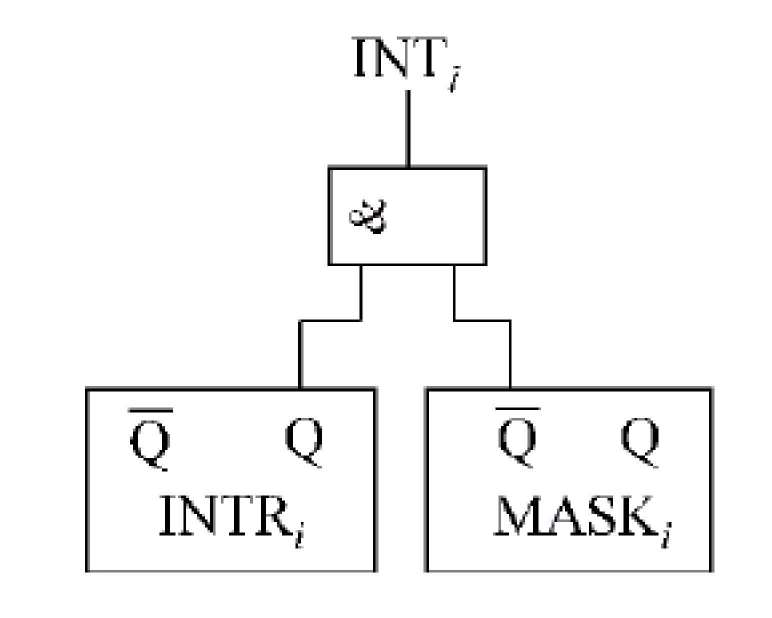


图 9-3 中断请求触发器和中断屏蔽触发器

在中断接口电路中,多个请求触发器组成一个请求寄存器,其内容称为中断字或中断码;多个屏蔽触发器组成一个屏蔽寄存器,其内容称为屏蔽字或屏蔽码。屏蔽字由程序来设置,其某一位的状态将成为该中断源能否真正发出中断请求信号的必要条件之一。这样,就可实现 CPU 对中断处理的控制,使中断能在系统中合理协调地进行。中断请求寄存器和中断屏蔽寄存器的作用如图 9-4 所示。具体地说,用程序设置的方法将屏蔽寄存器中的某一位置"1",则对应的中断请求被封锁,无法去参加排队判优;若屏蔽寄存器中的某一位置"0",才允许对应的中断请求送往 CPU。

有些中断请求是不可屏蔽的,也就是说,这些中断源的中断请求是不可被封锁的,一旦提出,CPU 必须立即响应。例如,电源掉电就是不可屏蔽中断。

12. 中断升级

通过改变中断屏蔽字可以改变中断优先级,使原级别较低的中断源变成较高的级别,这便称为中断升级。实际上,中断升级是一种动态改变优先级的方法。

这里所说的改变优先次序是指改变中断的处理次序。中断处理次序和中断响应次序是两个不同的概念,中断响应次序是由硬件排队电路决定的,无法改变。但是,系统软件根据需要可以改变屏蔽码,从而改变多重中断的中断处理次序。如果不使用屏蔽技术,中断处理次序就等于中断响应次序。

例如,某计算机的中断系统有 4 个中断源,每个中断源对应一个屏蔽码。表 9-1 为程序

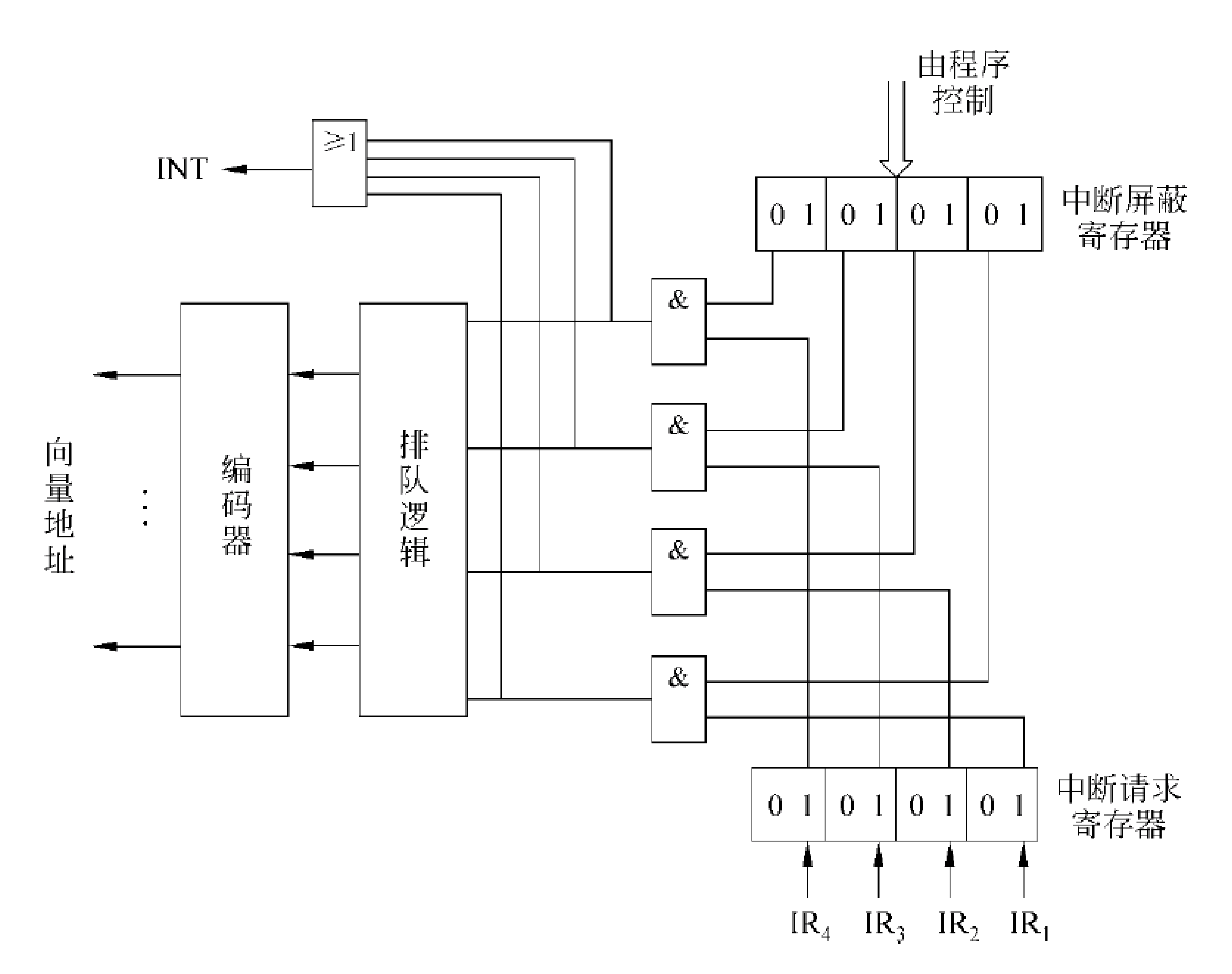


图 9-4　中断屏蔽寄存器的作用

优先级与屏蔽码的关系，中断响应的优先次序为 1→2→3→4（“0”表示开放，“1”表示屏蔽）。此时，中断的处理次序和中断的响应次序是一致的。

当多个中断请求同时出现时，处理次序与响应次序一致；当中断请求先后出现时，允许优先级别高的中断请求打断优先级别低的中断服务程序，实现中断嵌套。

在不改变中断响应次序的条件下，通过改写屏蔽码可以改变中断处理次序。例如，要使中断处理次序改为 2→1→4→3，则只需使中断屏蔽码改为如表 9-2 所示即可。

表 9-1　程序优先级与屏蔽码

程序级别	屏蔽码			
	1 级	2 级	3 级	4 级
第 1 级	1	1	1	1
第 2 级	0	1	1	1
第 3 级	0	0	1	1
第 4 级	0	0	0	1

表 9-2　改变处理次序的屏蔽码

程序级别	屏蔽码			
	1 级	2 级	3 级	4 级
第 1 级	1	0	1	1
第 2 级	1	1	1	1
第 3 级	0	0	1	0
第 4 级	0	0	1	1

中断屏蔽码可以是程序状态字 PSW 中的一个组成部分，程序的切换是通过交换新旧 PSW 实现的。如果中断处理次序改为 2→1→4→3，在同样中断请求的情况下，CPU 执行中断服务程序的次序将发生变化，图 9-5 为处理次序改变后的 CPU 运动轨迹。CPU 正在执行现行程序时，不能屏蔽任何中断请求，即 PSW 中的中断屏蔽码为 0000（处于全开放状态）。当中断请求①、②、④同时到来时，均进入排队器，显然 CPU 首先响应第①级中断请求，通过交换 PSW 实现程序切换，中断屏蔽码变为 1011。由于①级的中断屏蔽码是对②级是开放的，所以当①级的中断服务程序执行到开中断指令后，立即被②级中断请求打断，

CPU转去执行②级的中断服务程序,中断屏蔽码变为1111。待②级的中断服务程序执行完毕后,再返回接着执行①级中断服务程序,中断屏蔽码改变为1011。待第①级中断服务程序执行完毕,返回现行程序,中断屏蔽码又变成0000,此时才允许响应第④级中断请求,中断屏蔽码改变为0011。待④级的中断服务程序执行完毕后,最后返回现行程序,中断屏蔽码改变为0000。

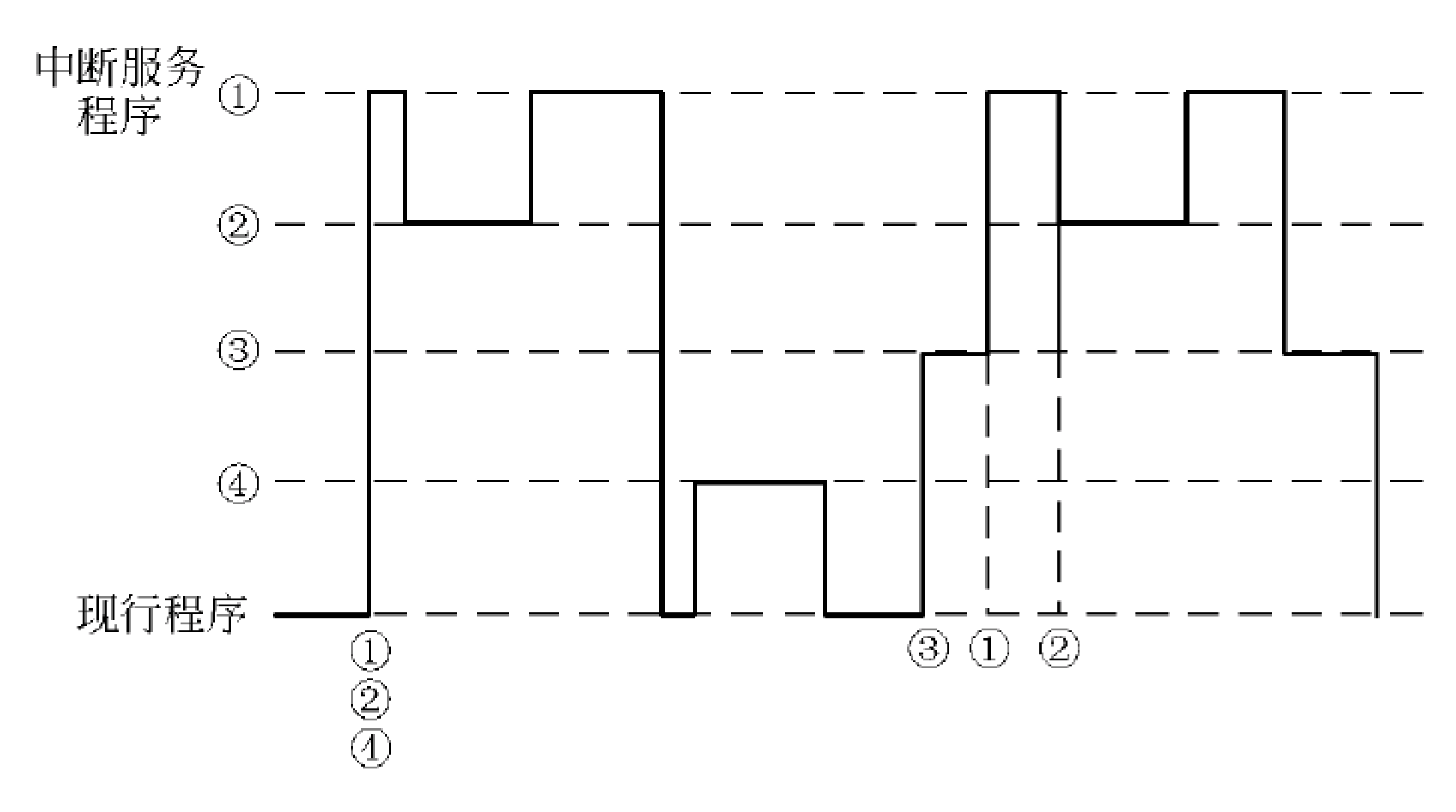


图9-5　处理次序改变后的CPU运动轨迹

在图9-5中,当第③级中断请求到来并在执行其中断服务程序的过程中,中断屏蔽码为0010。若又来了①级中断请求,③级中断服务程序将被①级中断请求打断,中断屏蔽码变为1011,转去执行①级中断服务程序。若在此过程中,又出现了②级中断请求,则转去执行②级的中断服务程序,中断屏蔽码变为1111。当②级的中断服务程序执行完毕,返回①级中断服务程序,中断屏蔽码为1011。当①级中断服务程序执行完毕,返回③级中断服务程序,中断屏蔽码为0010。最后返回现行程序,中断屏蔽码改变为0000。

由此可见,屏蔽技术向使用者提供了一种手段,即可用程序控制中断系统,动态地调度多重中断优先处理的次序,从而提高了中断系统的灵活性。

13. 中断全过程

中断全过程指的是从中断源发出中断请求开始,CPU响应这个请求,现行程序被中断,转至中断服务程序,直至中断服务程序执行完毕,CPU再返回原来的程序继续执行的整个过程。中断全过程可以分为5个阶段:中断请求、中断判优、中断响应、中断处理和中断返回。其中,中断处理就是执行中断服务程序,这是中断系统的核心。不同计算机系统的中断处理过程各具特色,但对多数计算机而言,其中断服务程序的流程如图9-6所示。

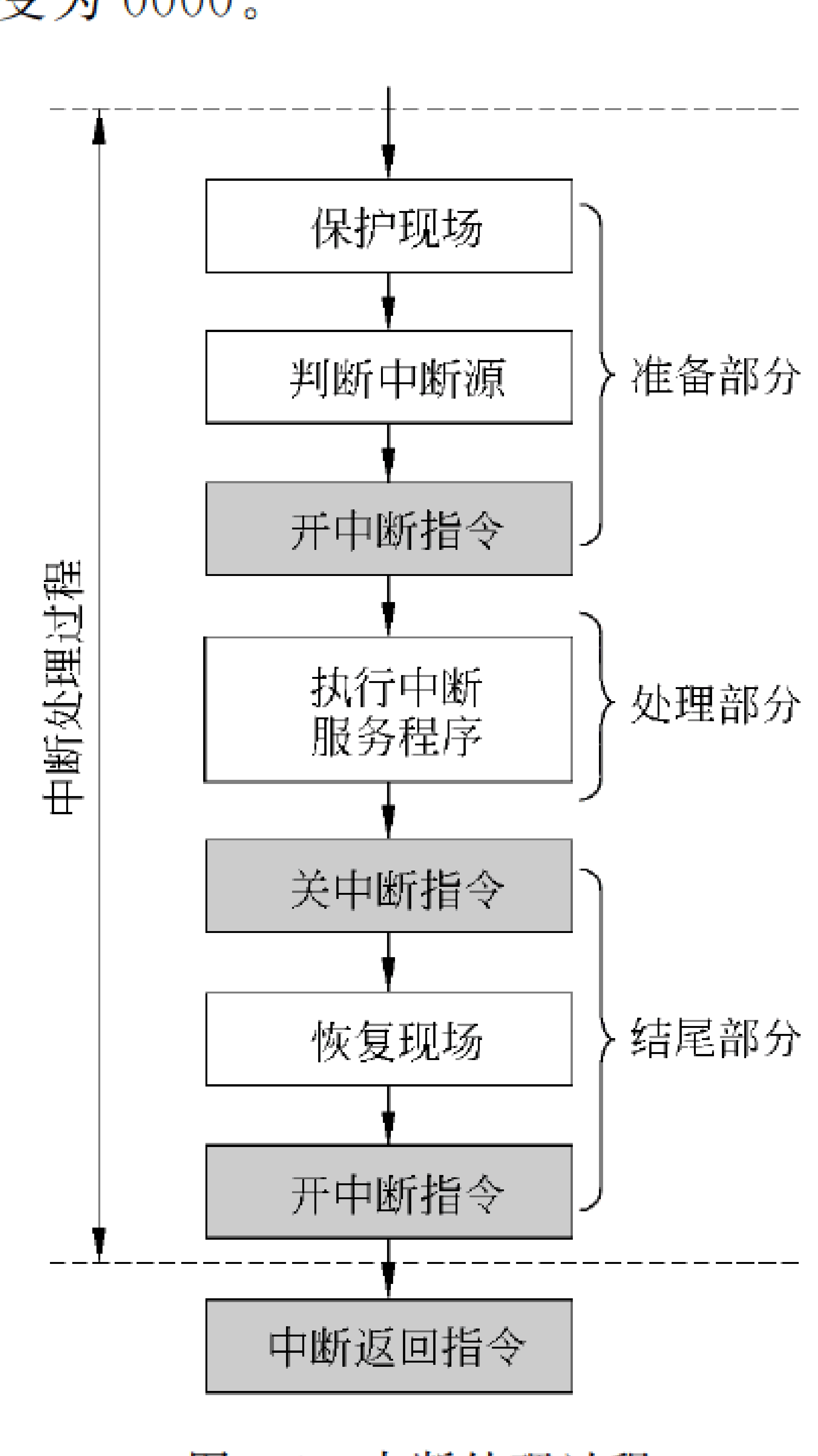


图9-6　中断处理过程

中断处理过程基本上由3个部分组成。第一部分为准备部分,首先要保护现场,对于非向量中断方式则还需要确定中断源,最后开中断,允许更高级的中断请求打断低级的中断服务程序;第二部分为处理部分,即

真正执行具体的为某个中断源服务的中断服务程序;第三部分为结尾部分,首先要关中断,以防止在恢复现场过程中被新的中断请求打断,接着恢复现场,然后开中断,以便返回原来的程序后可响应其他的中断请求。

图9-6中深色框中的功能由一条指令来完成,而浅色框中的功能可能需要多条指令来完成。如果系统仅允许单级中断,则图9-6中的第一个开中断指令和关中断指令可以取消。

中断服务程序的最后一条指令一定是中断返回指令。中断返回指令将完成恢复断点的功能,从而返回原来的程序继续执行。

多级中断和单级中断的区别在于多级中断在保护现场之后需要开中断,以便在执行某个中断服务程序的过程中可以响应级别更高的中断请求,而在恢复现场之前又要关中断,以保证恢复现场的过程中不能新的中断请求打断。多级中断的中断服务程序内应完成的任务有:①保存现场;②开中断;③中断事件处理;④关中断;⑤恢复现场;⑥开中断;⑦中断返回。

14. DMA方式的特点

DMA方式是在外设和主存之间开辟一条"直接数据通道",在不需要CPU干预也不需要软件介入的情况下,在两者之间进行的高速数据传送方式。在DMA传送方式中,对数据传送过程进行控制的硬件称为DMA控制器。当外设需要进行数据传送时,通过DMA控制器向CPU提出DMA传送请求,CPU响应之后将让出系统总线,由DMA控制器接管总线进行数据传送。DMA方式的主要特点是:

(1) 它使主存与CPU的固定联系脱钩,主存既可被CPU访问,又可被外设访问。

(2) 在数据块传送时,主存地址的确定、传送数据的计数等都用硬件电路直接实现。

(3) 主存中要开辟专用缓冲区,及时供给和接收外设的数据。

(4) DMA传送速度快,CPU和外设并行工作,提高了系统的效率。

(5) DMA在传送开始前要通过程序进行预处理,结束后要通过中断方式进行后处理。

DMA方式和程序中断方式的区别如表9-3所示。

表9-3 DMA方式和程序中断方式的区别

名　　称	程序中断方式	DMA方式
数据传送	由程序实现	由硬件实现
响应时间	指令周期结束	机器周期结束
处理异常情况	能	不能
中断请求	传送数据	传送结束释放总线
优先级	低	高

15. DMA控制器的组成

DMA控制器主要由以下几部分组成。

1) 主存地址计数器

用来存放主存中要交换数据的地址。该计数器的初始值为主存缓冲区的首地址,当DMA传送时,每传送一个数据,将地址计数器加1,从而以增量方式给出主存中要交换的一批数据的地址,直至这批数据传送完毕为止。

2）传送长度计数器

用来记录传送数据块的长度,其初始值为传送数据块的总字数或总字节数,每传送一个字或一个字节,计数器自动减1,当其内容为全0时表示数据块已全部传送完毕。

3）数据缓冲寄存器

用来暂存每次传送的数据。输入时,数据由外设(如磁盘)先送往数据缓冲寄存器,再通过数据总线送到主存。输出时,数据由主存通过数据总线送到数据缓冲寄存器,然后再送到外设。

4）DMA请求触发器

每当外设准备好数据后给出一个控制信号,使DMA请求触发器置位。

5）控制/状态逻辑

它由控制和时序电路以及状态标志等组成,用于指定传送方向,修改传送参数,并对DMA请求信号和CPU响应信号进行协调和同步。

6）中断机构

当一个数据块传送完毕后触发中断机构,向CPU提出中断请求,CPU将进行DMA传送的结尾处理。

16. DMA控制器的操作过程

以从外设传送一个数据块至主存为例,DMA控制器的操作如下:

(1) 首先,由外设向DMA控制器发出请求信号DREQ。

(2) DMA控制器向CPU发出总线请求信号HRQ。

(3) CPU向DMA控制器发出总线响应信号HLDA,此时DMA控制器获取了总线的控制权。

(4) DMA控制器向外设发出DMA响应信号DACK,表示DMA控制器已控制了总线,允许外设与主存交换数据。

(5) DMA控制器按主存地址计数器的内容发出地址信号作为主存地址的选择,同时主存地址计数器的内容加1(或减1)。

(6) DMA控制器发出IOR信号到外设,将外设数据读入总线,同时发出MEMW信号,将数据总线的数据写入地址总线选中的主存单元。

(7) 传送长度计数器减1。

重复步骤(5)、(6)、(7),直到字节计数器减到全0为止,数据块的DMA方式传送工作宣告完成。这时,DMA控制器的HRQ降为低电平,总线控制权交还CPU。

17. DMA传送方法

1）CPU停止访问主存法

用DMA请求信号迫使CPU让出总线控制权。CPU在现行机器周期执行完成之后,使其数据、地址总线处于三态,并输出总线响应信号。DMA控制器获得总线控制权以后,连续占用若干个存取周期(总线周期)进行成组连续的数据传送,直至批量传送结束,DMA控制器才把总线控制权交回CPU。在DMA传送期间,CPU处于保持状态,停止访问主存,仅能进行一些与总线无关的内部操作。图9-7(a)是这种传送方法的时间图。

这种方法的优点是控制简单,它适用于高速外设的成组传送;缺点是在DMA传送期间,主存的效能没有充分发挥,例如软盘读一个字节大约需要32μs,而RAM的存取周期只

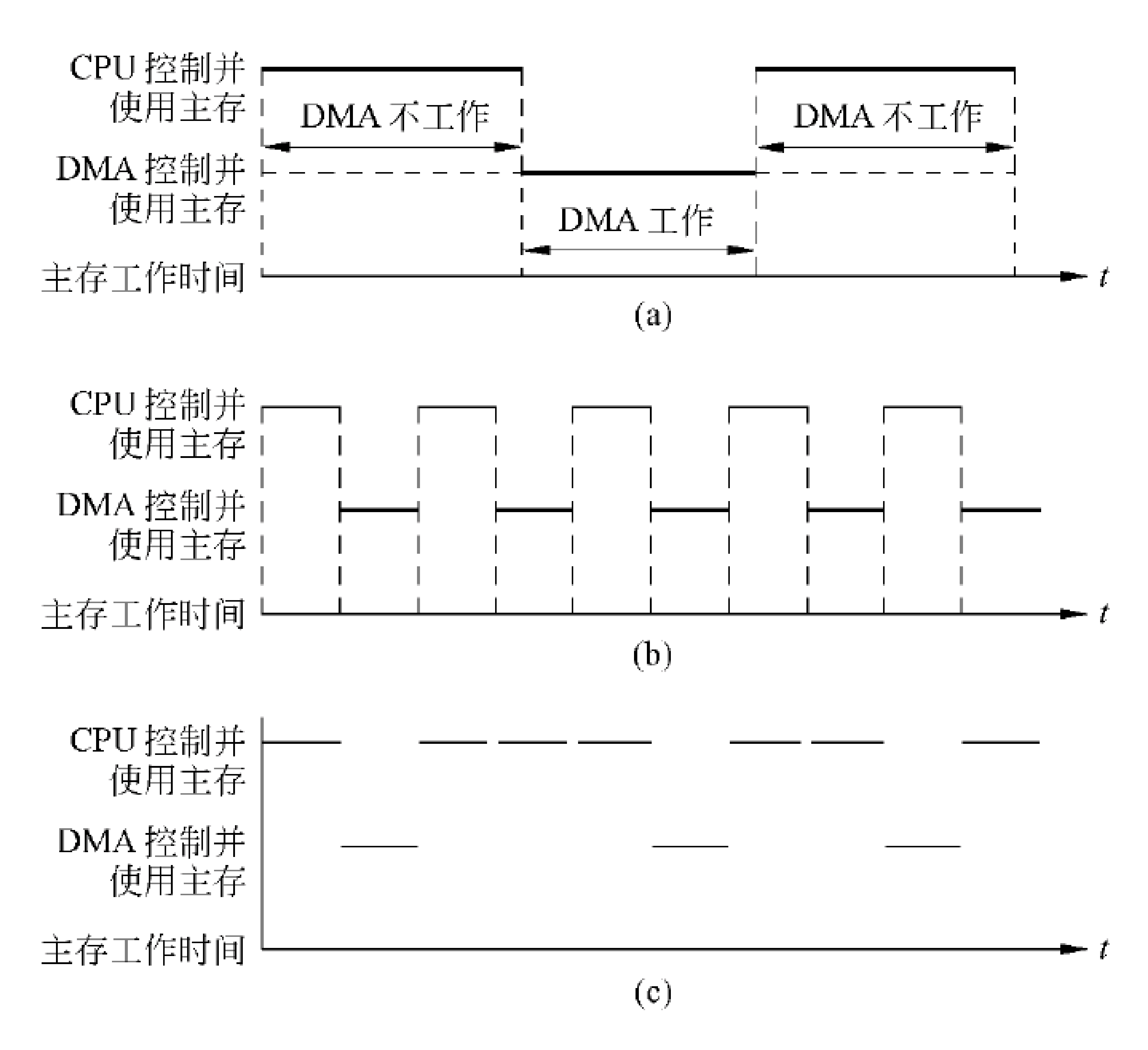


图 9-7　DMA 传送方法

有 1μs，那么就有 32μs－1μs＝31μs 的时间主存是空闲的，浪费较大。

2）存储器分时法

把原来的一个存取周期一分为二，一半由 CPU 使用，一半由 DMA 使用，使 CPU 和 DMA 交替地访问主存。这种方法无须申请和归还总线，使总线控制权的转移几乎不需要时间，所以对 DMA 传送来讲效率很高，而且 CPU 既不停止现行程序的运行也不进入保持状态，在 CPU 不知不觉中便进行了 DMA 传送，所以这种方法又称为“透明的 DMA”方法。但是，这种方法需要主存在原来的存取周期内为两个部件服务，如果要维持 CPU 的访存速度不变，就要求主存的工作速度提高一倍。另外，由于大多数外设的速度都不能与 CPU 相匹配，所以供 DMA 使用的时间片可能成为空操作，将会造成一些不必要的浪费。图 9-7(b)是这种方法的时间图。

3）周期挪用法

周期挪用又称周期窃取，是前两种方法的折中。当外设没有 DMA 请求时，CPU 按程序要求访问主存；一旦外设有 DMA 请求并且获得 CPU 响应后，CPU 让出一个周期的总线控制权，由 DMA 控制器控制系统总线，挪用一个存取周期进行一次数据传送，传送一个字或一个字节；然后，DMA 控制器将总线控制权交回 CPU，CPU 继续进行自己的操作，等待下一个 DMA 请求的到来。图 9-7(c)就是这种方法的时间图。周期挪用法适用于 I/O 设备读写周期大于主存存储周期的情况，若 DMA 传送期间 CPU 无须访存，则周期挪用对 CPU 执行程序无任何影响。

18. DMA 传送过程

DMA 的数据传送过程可分为 3 个阶段：预处理阶段、传送阶段和后处理阶段。

1）DMA 预处理

在 DMA 传送之前必须要做准备工作，即初始化，这是由 CPU 来完成的。CPU 首先执

行几条 I/O 指令,用于测试外设的状态、向 DMA 控制器的有关寄存器设置初值、设置传送方向、启动该外部设备等。

在这些工作完成之后,CPU 继续执行原来的程序,在外设准备好发送的数据(输入)或接收的数据已处理完毕(输出)时,外设向 DMA 控制器发 DMA 请求,再由 DMA 控制器向 CPU 发总线请求。

2) 数据传送

DMA 的数据传送可以以单字(或字节)为基本单位,也可以以数据块为基本单位。在数据传送阶段,由 DMA 控制器自动地完成外设与主存之间的数据传送,DMA 控制器所做的工作如图 9-8 所示。

需要特别指出的是,此时 CPU 可以继续执行原来的程序,不需要插手任何与数据传送相关的工作。

3) DMA 后处理

当传送长度计数器计到全 0 时,DMA 传送结束,DMA 控制器向 CPU 发中断请求,CPU 停止原来程序的执行,转去执行中断服务程序,做 DMA 结束处理工作。

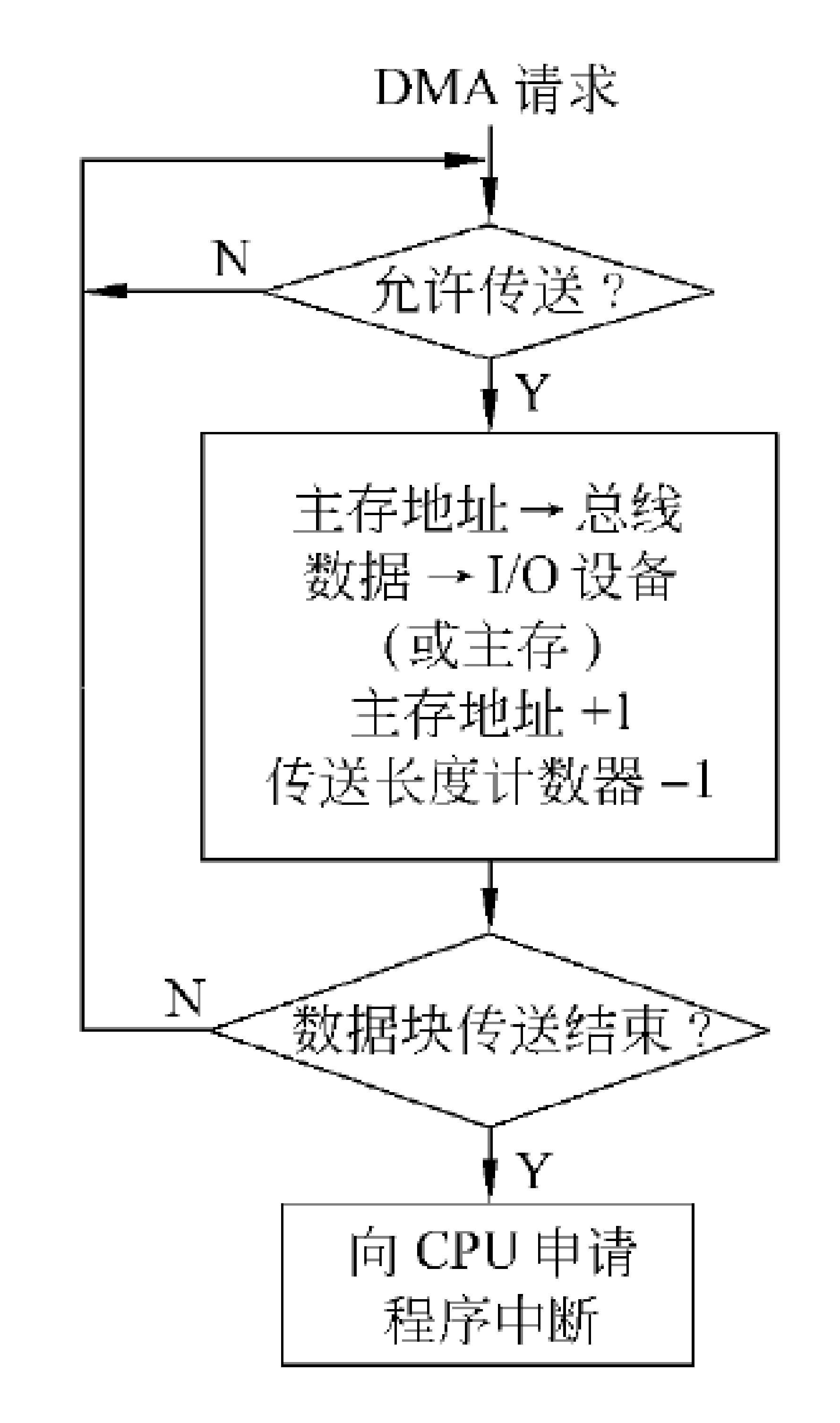


图 9-8 DMA 控制器所做的工作

19. 通道控制方式与 DMA 方式的比较

DMA 和通道控制方式最基本的相同点是从 CPU 中接管外设与主存交换数据过程的控制权,使外设能与主机并行工作。它们之间主要的不同之处在于:

(1) DMA 与通道的工作原理不同。DMA 通过专门设计的硬件控制逻辑来控制数据交换的过程;而通道则是一个具有特殊功能的处理器,它具有自己的指令和程序,通过执行通道程序来控制数据交换的过程。

(2) DMA 与通道的功能不同。通道是在 DMA 的基础上发展起来的,因此通道要比 DMA 的功能更强。

(3) DMA 与通道所控制的外设类型不同。DMA 只能控制速度较快、类型单一的外设;而通道则可以支持多种类型的外设。

20. 通道类型

根据通道的工作方式,可将通道分为字节多路通道、选择通道和数组多路通道 3 种类型。

1) 字节多路通道

字节多路通道是一种简单的共享通道,用于连接与管理多台低速设备,以字节交叉方式传送信息。

一个字节多路通道,包括多个按字节方式传送信息的子通道。每个子通道服务于一个设备控制器,每个子通道都可以独立地执行通道程序。各个子通道可以并行工作,但是,所有子通道的控制部分是公共的,各个子通道可以分时地使用。

字节多路通道的实际流量是连接在这个通道上的所有设备的数据传输速率之和,若通道上接有 P 台设备,则字节多路通道的实际流量为:

$$f_{\text{byte}} = \sum_{i=1}^{P} f_i$$

2）选择通道

选择通道又称高速通道，在物理上它也可以连接多个设备，但这些设备不能同时工作，在一段时间内通道只能选择一台设备进行数据传送，此时该设备可以独占整个通道。因此，选择通道一次只能执行一个通道程序，只有当这个设备的通道程序全部执行完毕后，才能执行其他设备的通道程序。

选择通道主要用于连接高速外设（如磁盘、磁带等），信息以成组方式高速传送。但是，在数据传送过程中还有一些辅助操作（如磁盘机的寻道等），此时会使通道处于等待状态，所以虽然选择通道具有很高的数据传送速率，但是整个通道的利用率并不高。

选择通道的实际流量等于连接在通道上的所有设备中数据传输率最高者，若通道上接有 P 台设备，则选择通道的实际流量为：

$$f_{\text{select}} = \max_{i=1}^{P} f_i$$

3）数组多路通道

数组多路通道是把字节多路通道和选择通道的特点结合起来的一种通道结构。它的基本思想是：当某设备进行数据传送时，通道只为该设备服务；当设备在执行寻址等辅助操作时，通道暂时断开与这个设备的连接，挂起该设备的通道程序，转去为其他设备服务，即执行其他设备的通道程序。

数组多路通道有多个子通道，既具有多路并行操作的能力，又具有很高的数据传送速率，使通道的效率充分得到发挥。

数组多路通道的实际流量等于连接在通道上的所有设备中数据传输率最高者，若通道上接有 P 台设备，则数组多路通道的实际流量为：

$$f_{\text{block}} = \max_{i=1}^{P} f_i$$

为了使通道所接外部设备在满负荷工作时仍然不丢失信息，应使通道的实际最大流量不能超过通道的极限流量。

如果在 I/O 系统中有多个通道，各个通道是并行工作的，则 I/O 系统的极限流量应当是各通道或各子通道工作时的极限流量之和。

9.3 典型例题详解

【例 9.1】 主机与外设间的信息交换通过访问与外设相对应的寄存器（端口）来实现，对这些端口的编址方式有几种？它们各有哪些优缺点？80x86 微机采用的是哪一种方式？它的 I/O 地址空间可以直接寻址和间接寻址，它们各自最大可以提供多少个 8 位端口、16 位端口或 32 位端口？

解： I/O 端口编址方式有统一编址和独立编址两种。这两种方式各有优缺点，它们的比较如表 9-4 所示。

表 9-4　两种 I/O 端口编址方式比较

	独立编址方式	统一编址方式
优点	I/O 指令和访存指令容易区分，外设地址线少，译码简单，主存空间不会减少	总线结构简单，全部访存类指令都可用于控制外设，可直接对外设寄存器进行各种运算
缺点	控制线增加了 I/O 读和 I/O 写信号	占用主存一部分地址，缩小了可用的主存空间

80x86 微机采用独立编址方式。直接寻址 I/O 端口的寻址范围为 00～FFH，至多为 256 个端口地址。这时程序可以指定：256 个 8 位端口或 128 个 16 位端口或 64 个 32 位端口。间接寻址由 DX 寄存器间接给出 I/O 端口地址，DX 寄存器长 16 位，至多有 65 536 个端口地址。这时程序可指定：65 536 个 8 位端口或 32 768 个 16 位端口或 16 384 个 32 位端口。

【例 9.2】 主机和外设之间的信息传送控制方式有哪几种？它们各有哪些特点？各适用于什么场合？试说明 CPU 利用程序查询方式从键盘读入一个数据的工作过程。

解：主机和外设之间的信息传送控制方式有以下几种：

(1) 程序查询方式。这是一种程序直接控制方式，输入和输出完全是通过 CPU 执行程序来完成的。一旦某一外设被选中并启动之后，主机将查询这个外设的某些状态位，看其是否准备就绪。若外设未准备就绪，主机将再次查询；若外设已准备就绪，则执行一次 I/O 操作。这种方式控制简单，但系统效率很低，仅适用于外设的数目不多，对 I/O 处理的实时要求不那么高，CPU 的操作任务比较单一、并不很忙的情况。

(2) 程序中断方式。主机接到外设的中断请求后，暂时中止原来执行的程序，转去执行中断服务程序对外部请求进行处理，在中断处理完毕后返回原来的程序继续执行。显然，程序中断不仅适用于外部设备的输入输出操作，也适用于对外界发生的随机事件的处理。

程序中断允许主机和外设同时并行工作，但是完成一次程序中断需要许多辅助操作，对于一些高速外设，由于信息交换是成批的，如果处理不及时，可能会造成信息丢失，因此，它主要适用于中、低速外设。

(3) 直接存储器存取(DMA)方式。DMA 方式是在主存和外设之间开辟直接的数据通路，可以进行基本上不需要 CPU 介入的主存和外设之间的信息传送，这样不仅能保证 CPU 的高效率，而且能满足高速外设的需要。

(4) I/O 通道控制方式。这是 DMA 方式的进一步发展，在系统中设有通道控制部件，每个通道上挂有若干个外设，主机在执行 I/O 操作时，只需启动有关通道，通道将执行通道程序，从而完成 I/O 操作。

CPU 利用程序查询方式从硬盘上读取一个数据的过程是：CPU 首先启动键盘工作，然后测试键盘状态，若键盘数据未准备就绪，则输入缓冲寄存器的内容不可以使用，继续查询；若键盘数据已准备就绪，则执行输入指令取走该数据。CPU 从键盘读取一个数据的过程如图 9-9 所示。

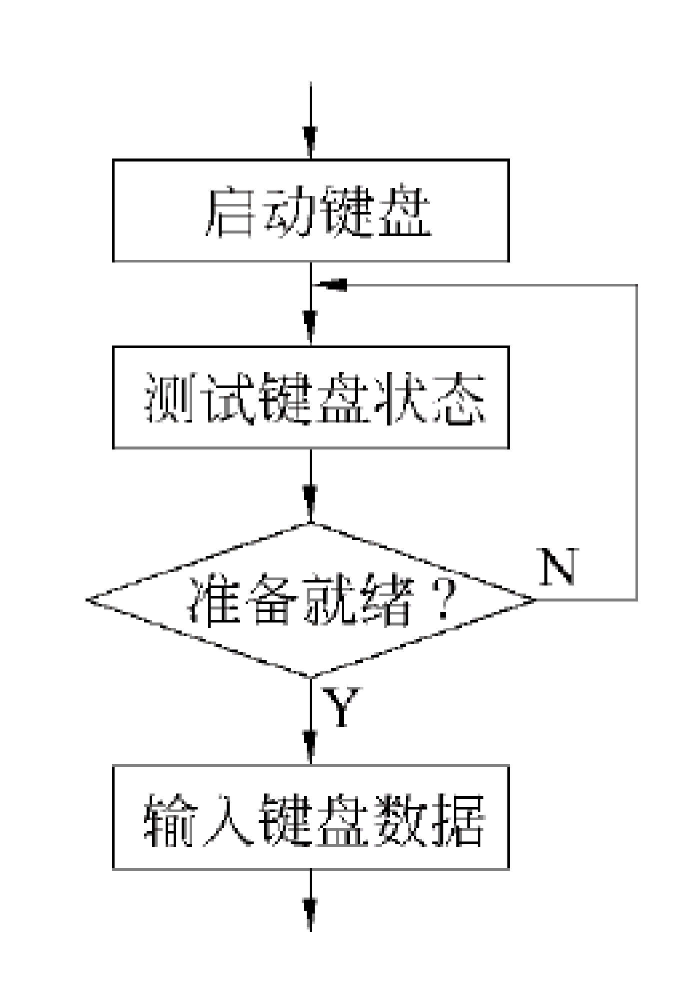


图 9-9　CPU 从键盘读取一个数据的过程

【例 9.3】 在程序查询方式的输入输出系统中，假设不考虑处理时间，每一个查询操作需要 100 个时钟周期，CPU 的时钟频率为 50MHz。现有鼠标和硬盘两个设备，而且 CPU 必须

每秒对鼠标进行 30 次查询，硬盘以 32 位字长为单位传输数据，即每 32 位被 CPU 查询一次，传输率为 2MB/s。求 CPU 对这两个设备查询所花费的时间比率，由此可得出什么结论？

解：CPU 每秒对鼠标进行 30 次查询，所需得时钟周期数为 $100\times30=3000$，根据 CPU 的时钟频率为 50MHz，故对鼠标的查询占用 CPU 的时间比率为：

$$\frac{3000}{50\times10^6}\times100\%=0.006\%$$

对于硬盘，每 32 位被 CPU 查询一次，每秒查询次数为 $2\text{MB}\div4\text{Byte}=512\text{K}$，则每秒查询的时钟周期数为：

$$100\times512\times1024=52.4\times10^6$$

对磁盘的查询占用 CPU 的时间比率为：

$$\frac{52.4\times10^6}{50\times10^6}\times100\%=105\%$$

以上结果表明，对鼠标的查询基本不影响 CPU 的性能，而即使 CPU 将全部时间都用于对磁盘的查询也不能满足磁盘传输的要求，所以 CPU 一般不采用程序查询方式与磁盘交换信息。

【例 9.4】 设有 8 个中断源 INT_1～INT_8，用软件方式排队判优。

(1) 设计中断请求逻辑电路。

(2) 如何判别中断源？画出软件判优的流程。

解：(1) 中断请求逻辑电路如图 9-10 所示。

用软件方式排队判优，所需硬件非常简单，只需一个或门和一个存放 8 个请求信号的寄存器即可。根据或门的输出判别有无中断请求。若有，再通过查询判优程序对寄存器对应位进行检测。

(2) 软件判优程序中，检测顺序是按优先级的大小排列的，最先检测的中断源具有最高的优先级，其次检测的中断源具有次高优先级，如此下去，最后检测的中断源具有最低的优先级。软件查询判优的流程如图 9-11 所示。

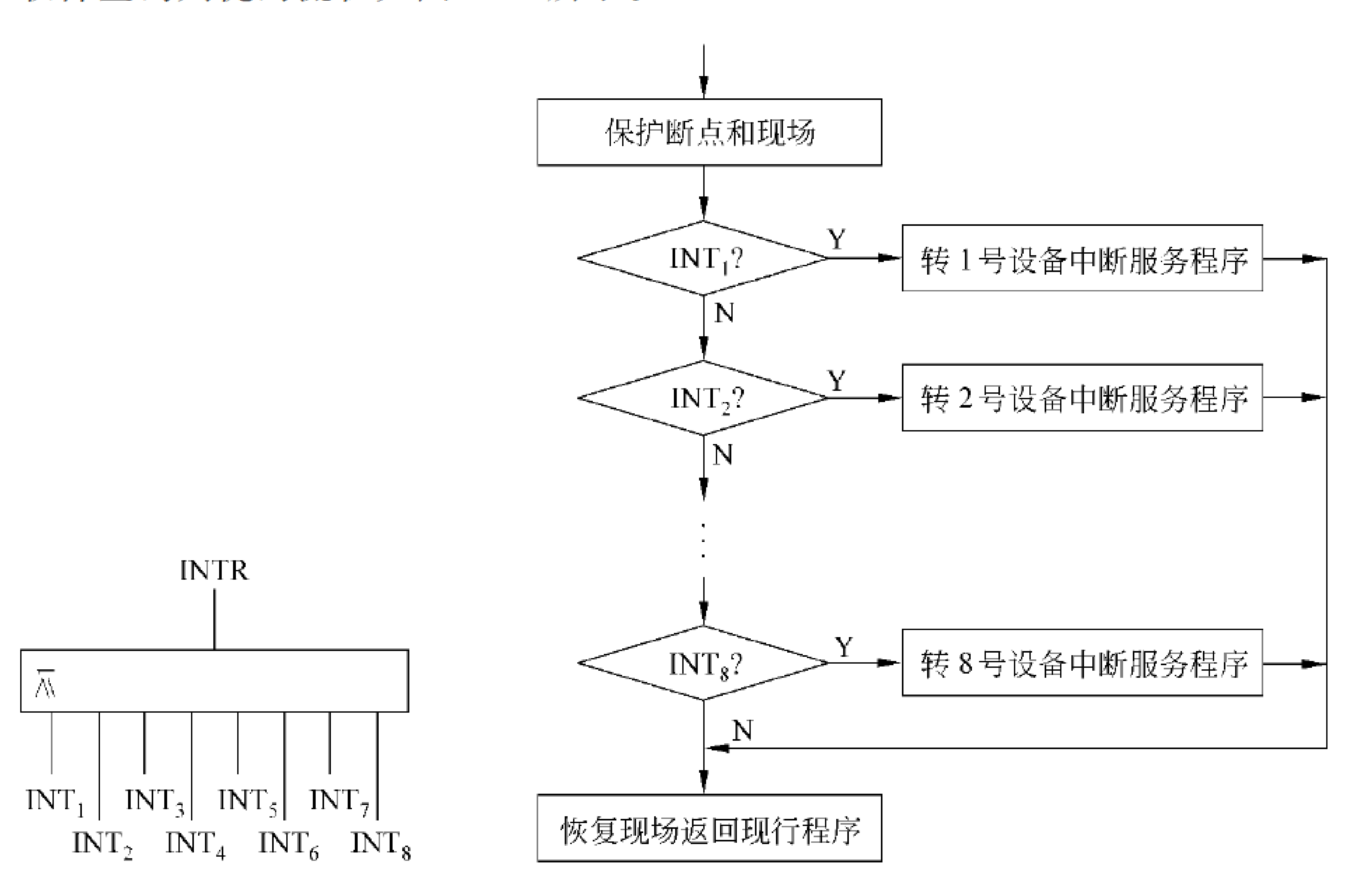


图 9-10 中断请求逻辑电路

图 9-11 软件查询判优流程

软件判优方法简单,可以灵活地修改中断源的优先级别;但是,查询、判优完全是靠程序实现的,不但占用CPU时间,而且判优速度慢。

【例9.5】 图9-12是从实时角度观察到的中断嵌套。试问:这个中断系统可实现几重中断?请分析图9-12中的中断过程。

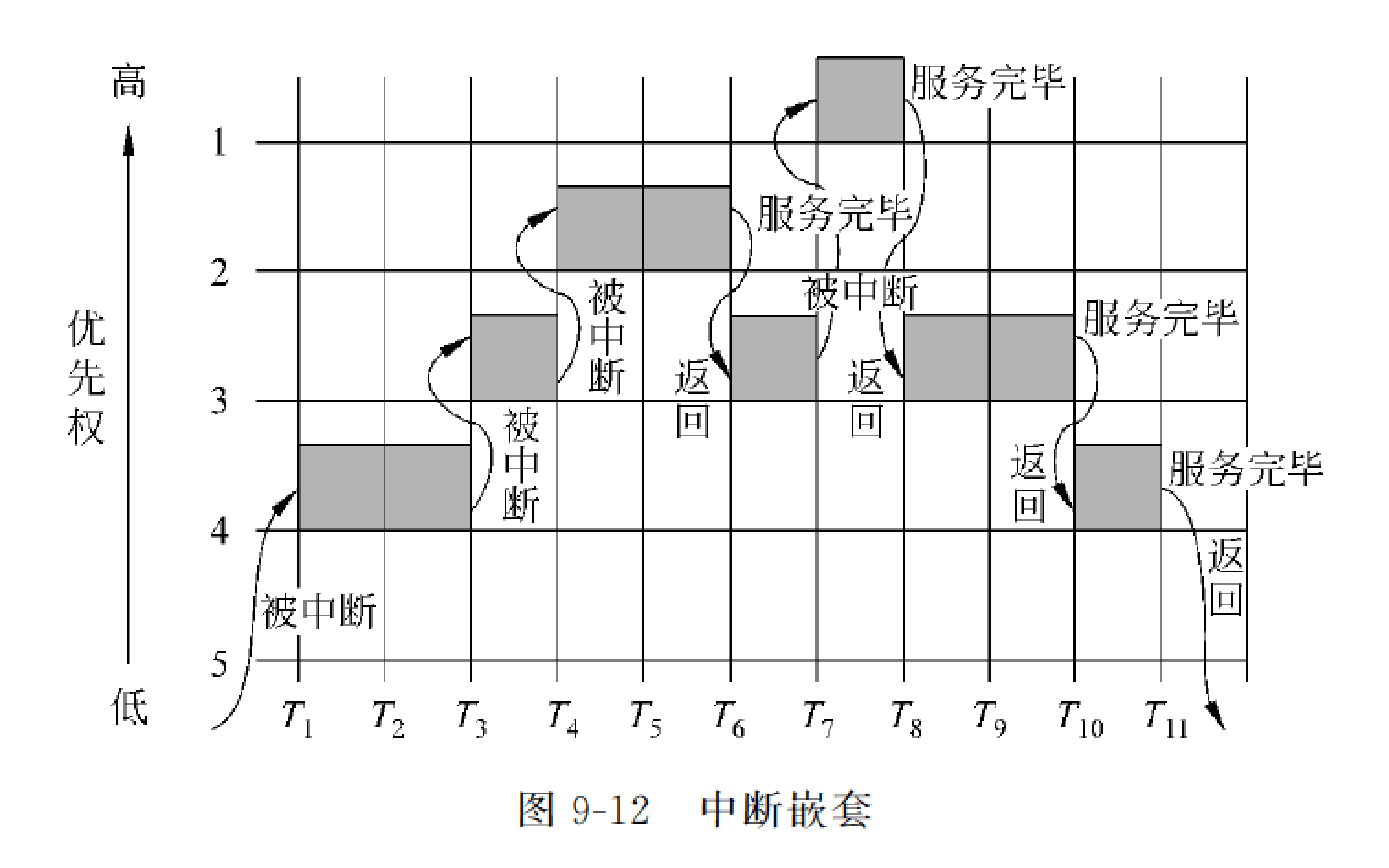


图9-12 中断嵌套

解:该中断系统可以实现5重中断。中断优先级的顺序是,优先权1最高,现行程序运行于最低优先权(优先权为6)。图9-12中出现了4重中断,其中断过程如下:

现行程序运行到T_1时刻,响应优先权4的中断源的中断请求并进行中断服务。到T_3时刻,优先权4的中断服务还未结束,但又出现了优先权3的中断源的中断请求,暂停优先权4的中断服务,而响应优先权3的中断。到T_4时刻,又被优先权2的中断源所中断,直至T_6时刻,返回优先权3的中断服务。到T_7时刻,优先权1的中断源发出中断请求并被响应,到T_8时刻优先权1中断服务完毕,返回优先权3的服务程序。到T_{10}时刻优先权3中断服务结束,返回优先权4的中断服务。到T_{11}时刻优先权4的中断服务结束,最后返回现行程序。在图9-12中,优先权3的中断服务程序被中断2次,而优先权5的中断请求没有发生。

【例9.6】 某中断系统可以实现5重中断,中断优先级的顺序是1→2→3→4→5(其中优先权1最高)。

若现行程序运行到T_1时刻,响应优先权4的中断源的中断请求;在此中断处理尚未结束的T_2时刻,又出现了优先权3的中断源的中断请求;当优先权3未处理结束的T_3时刻,又出现了优先权2的中断源的中断请求;待优先权2的中断处理完毕刚一返回的T_4时刻,又被优先权1的中断源的中断请求打断。请从实时角度画出观察到的CPU运动轨迹(从现行程序被中断直至返回现行程序止),在图中标出中断请求和返回点,并简要说明。

解:CPU运动轨迹如图9-13所示。

T_1时刻响应④级中断请求并进行中断服务,到T_2时刻来了更高级的中断请求③,④级中断服务程序被打断,转③级中断服务。到T_3时刻,又来了②级中断请求,③级中断服务程序被打断,转②级中断服务。②级中断服务程序执行完毕返回③级中断服务时,又来了①级中断请求,故先执行①级中断服务程序。待①级中断服务程序执行完毕,返回③级中断服务程序。待③级中断服务程序执行完毕,返回④级中断服务程序,最后返回现行程序。

【例9.7】 有5个中断源D_1、D_2、D_3、D_4和D_5,它们的中断优先级从高到低分别是1级、

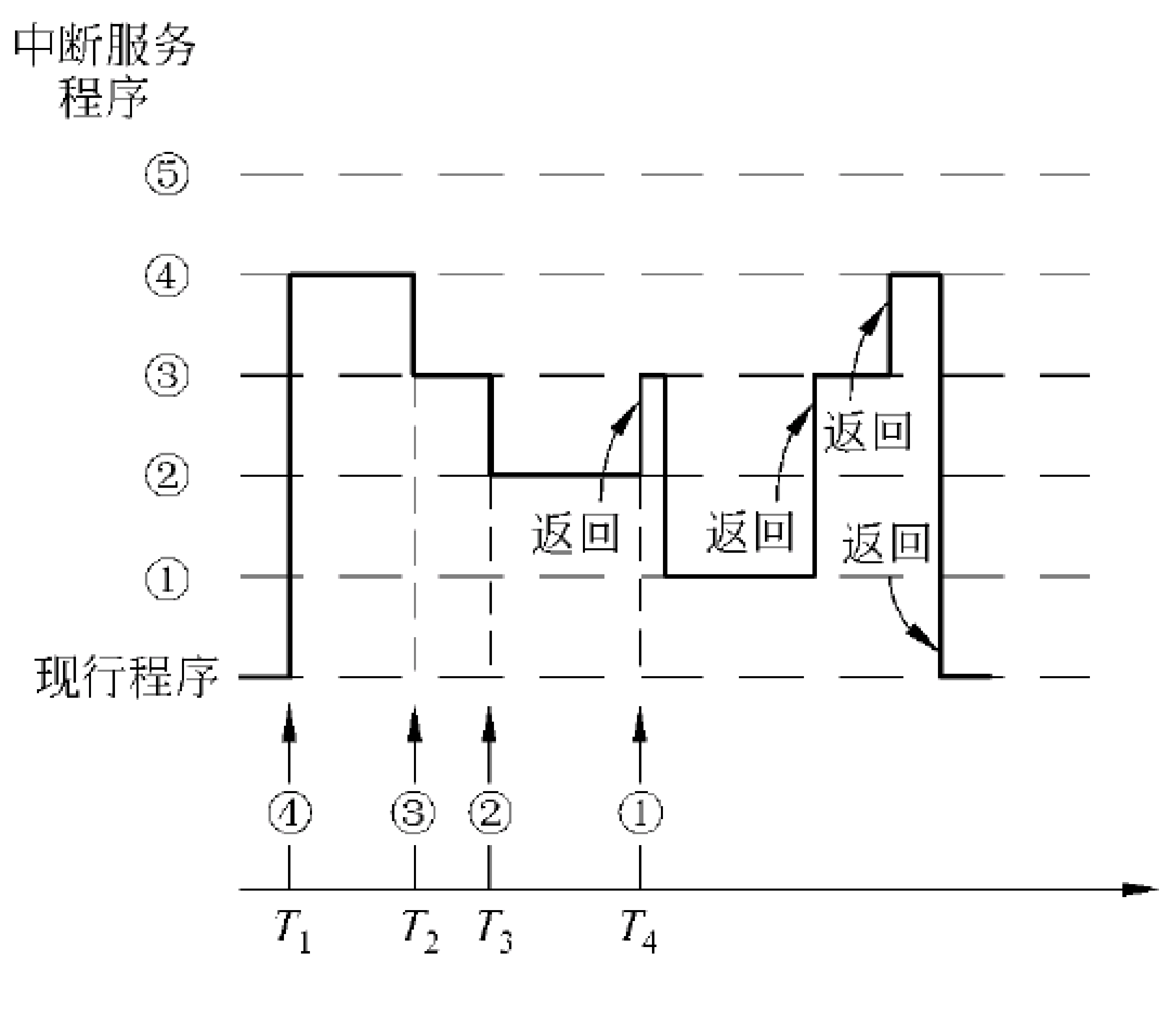


图 9-13　CPU 运动轨迹

2 级、3 级、4 级和 5 级。这些中断源的中断优先级、正常情况下的中断屏蔽码和改变后的中断屏蔽码如表 9-5 所示。每个中断源有 5 位中断屏蔽码，其中，“0”表示该中断源开放，“1”表示该中断源被屏蔽。

表 9-5　中断屏蔽码

中　断　源	中断源优先级	正常的中断屏蔽码					改变后的中断屏蔽码				
		D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5
D_1	1	1	1	1	1	1	1	0	0	0	0
D_2	2	0	1	1	1	1	1	1	0	0	0
D_3	3	0	0	1	1	1	1	1	1	0	0
D_4	4	0	0	0	1	1	1	1	1	1	1
D_5	5	0	0	0	0	1	1	1	1	0	1

(1) 当使用正常的中断屏蔽码时，处理机响应各中断源的中断请求的先后次序是什么？实际上中断处理的先后次序是什么？

(2) 当使用改变后的中断屏蔽码时，处理机响应各中断源的中断请求的先后次序是什么？实际上中断处理的先后次序是什么？

(3) 如果采用改变后的中断屏蔽码，D_1、D_2、D_3、D_4 和 D_5 这 5 个中断源同时请求中断时，画出处理机响应中断源的中断请求和实际运行中断服务程序过程的示意图。

解：(1) 当使用正常的中断屏蔽码时，中断响应次序是 1→2→3→4→5，中断处理次序也是 1→2→3→4→5。

(2) 当使用改变后的中断屏蔽码时，中断响应次序不变，中断处理次序是 4→5→3→2→1。

(3) 处理机响应中断源的中断请求和实际运行中断服务程序过程的示意图如图 9-14 所示。

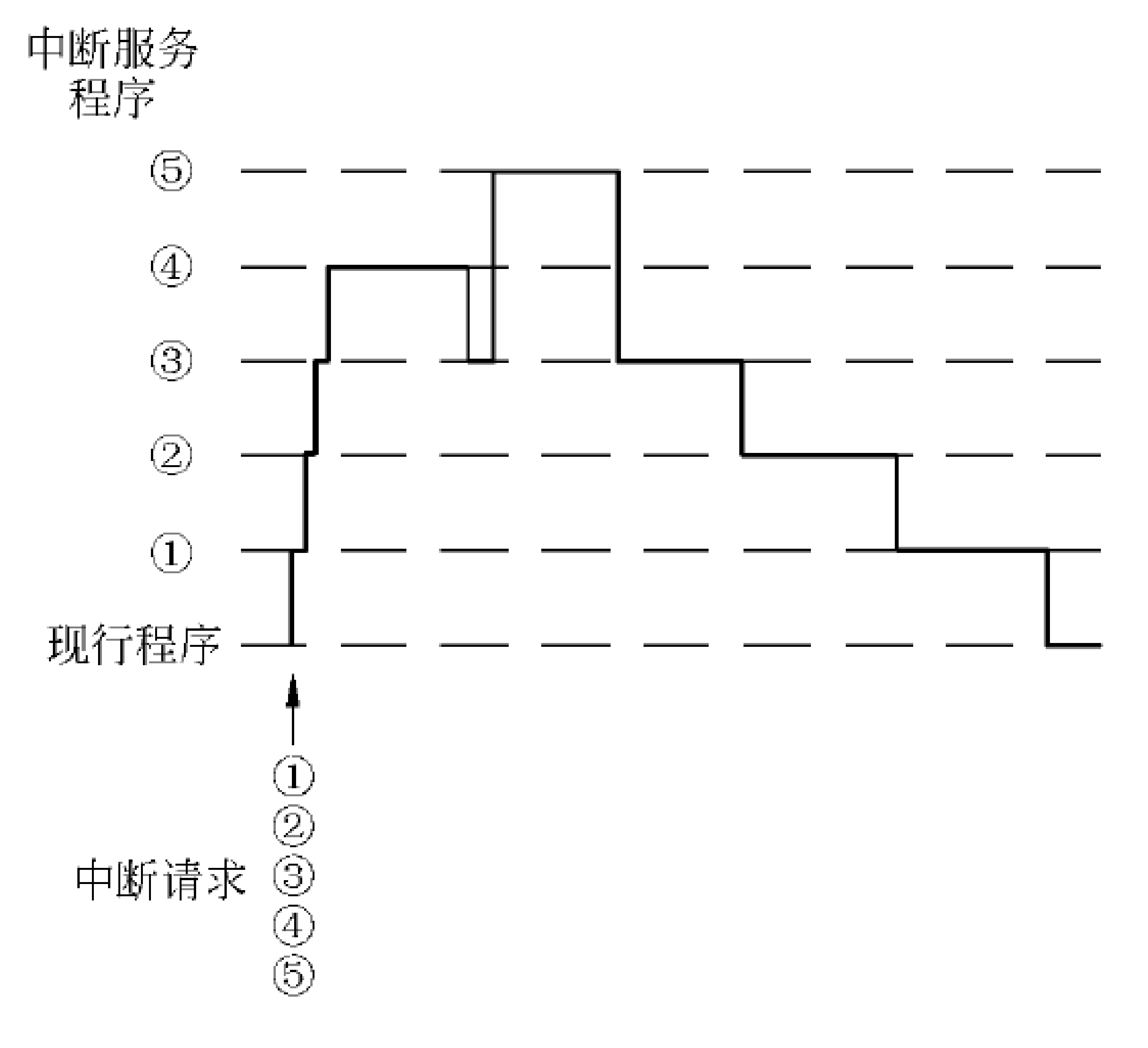


图 9-14　处理机运行示意图

【例 9.8】 在一个 8 级中断系统中,硬件中断响应从高到低优先顺序是 1→2→3→4→5→6→7→8,设置中断屏蔽寄存器后,中断处理的优先顺序变为 1→3→5→7→2→4→6→8。如果 CPU 在执行一个应用程序时有 5、6、7、8 级 4 个中断同时到达,CPU 在按优先顺序处理到第 3 个中断请求的过程中又有一个 3 级中断请求到达 CPU,试画出 CPU 响应这些中断的顺序示意图。

解:CPU 响应中断的顺序示意图如图 9-15 所示。

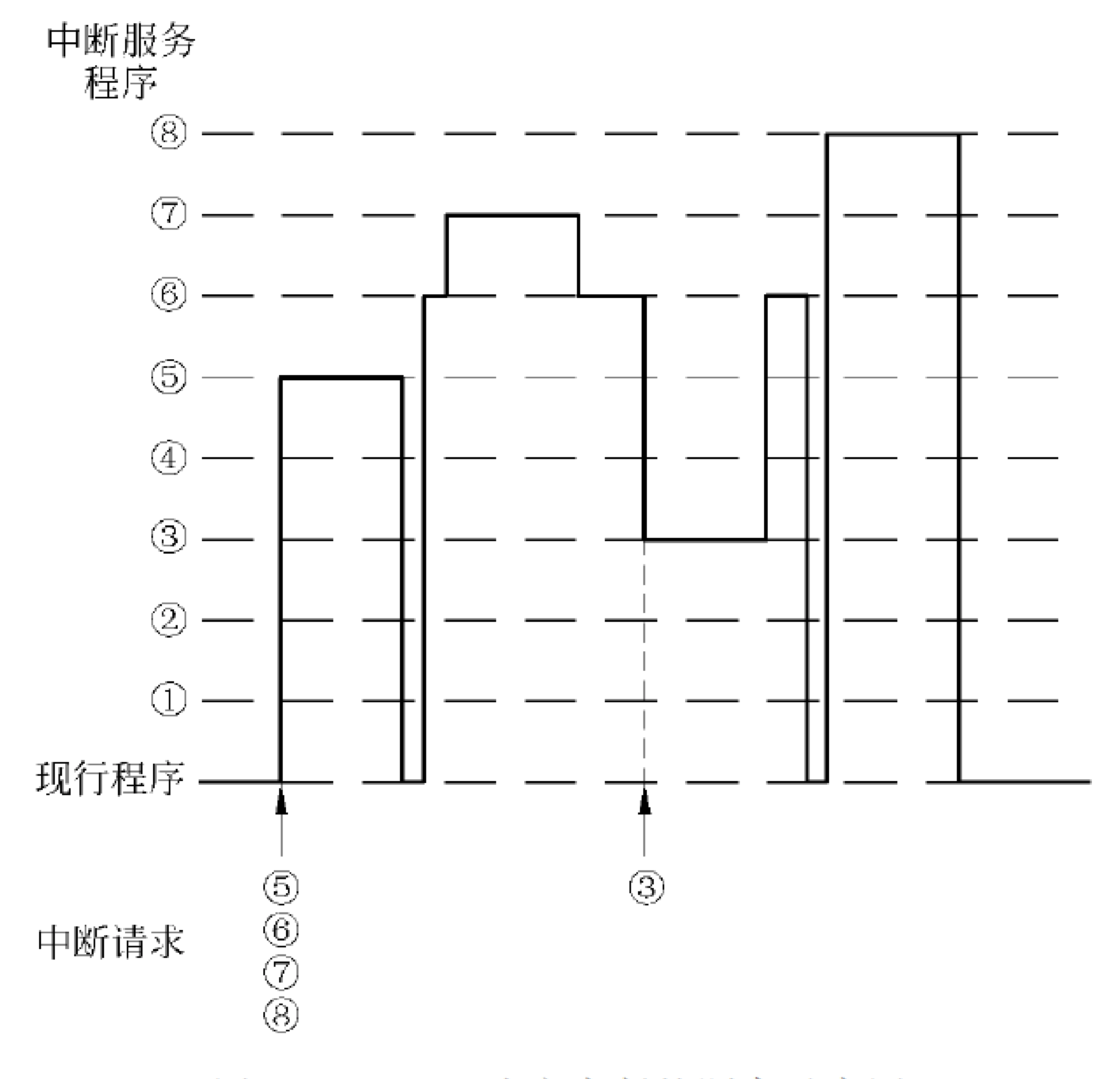


图 9-15　CPU 响应中断的顺序示意图

【例 9.9】 设某计算机有 4 级中断 A、B、C、D,其硬件排队优先级次序为 A>B>C>D。表 9-6 列出了执行每级中断服务程序所需的时间。

表 9-6　中断服务程序所需的时间

中断服务程序	所需时间	中断服务程序	所需时间
A	5μs	C	3μs
B	15μs	D	12μs

如果想以执行中断服务程序的时间作为确定中断优先级的尺度：时间越短优先级越高。

(1) 请指出如何为各级中断服务程序设置屏蔽码?

(2) 如果 A、B、C、D 分别在 6μs、8μs、10μs、0μs 时刻发出中断请求，请画出 CPU 执行中断服务程序的序列。

(3) 基于上题，请计算上述 4 个中断服务程序的平均执行时间。

解：(1) 中断服务程序屏蔽码如表 9-7 所示。

表 9-7　例 9.9 的中断屏蔽码

中断源	中断屏蔽码			
	A	B	C	D
A	1	1	0	1
B	0	1	0	0
C	1	1	1	1
D	0	1	0	1

(2) 各级中断源发出的中断请求信号的时刻，画出 CPU 执行中断服务程序的序列，如图 9-16 所示。

中断处理的优先级别是 C>A>D>B。0μs 时，D 请求来到，由于没有其他的中断请求，所以开始执行中断服务程序 D。第 6μs 时，A 请求来到，A 的优先级高于 D，转去执行中断服务程序 A。第 8μs 时，B 请求来到，由于 B 的优先级低于 A，所以不响应 B 请求，继续执行中断服务程序 A。第 10μs 时，C 请求来到，C 的优先级最高，虽然此时中断服务程序 A 还没有结束，也必须暂停转去执行中断服务程序 C。中断服务程序 C 所需时间为 3μs，当第 13μs 时，中断服务程序 C 执行完毕，返回执行中断服务程序 A。第 14μs 时，中断服务程序 A 执行完毕(总共执行时间 5μs)，返回执行中断服务程序 D。第 20μs 时中断服务程序 D 执行完毕(总共执行时间 12μs)，返回现行程序。因为 B 请求还存在，所以此时开始执行中断服务程序 B，直至第 35μs 时结束(总共执行时间 15μs)。

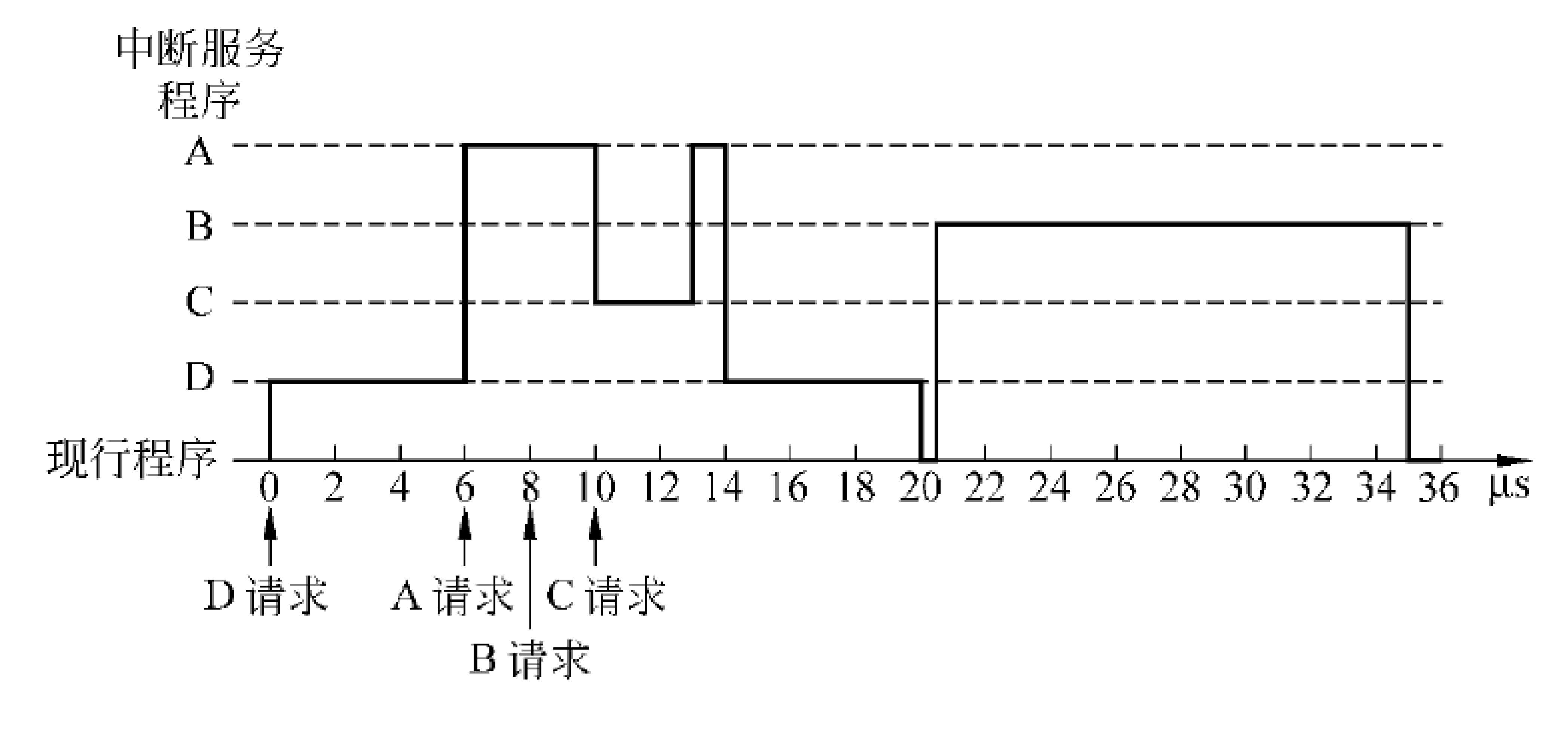


图 9-16　CPU 执行中断服务程序的序列

(3) 由于在 35μs 时间内，完成了 4 级中断的处理，所以平均执行时间 = 35 ÷ 4 = 8.75μs。

【例 9.10】　从中断源的急迫程度、CPU 响应时间和接口控制电路 3 个方面，说明程序

中断和 DMA 方式的差别。

解:(1) 从中断源的急迫程度来看,DMA 高于中断。

(2) 从 CPU 响应时间来看,CPU 对中断的响应是在执行完一条指令之后,而对 DMA 的响应则可以在指令执行过程中的任何一个机器周期结束时进行。

(3) 从接口控制电路来看,DMA 控制器包括的以下部件:

① 主存地址计数器;

② 传送长度计数器;

③ 数据缓冲寄存器;

④ DMA 请求触发器;

⑤ 控制/状态逻辑;

⑥ 中断机构。

中断接口控制电路包括以下部件:

① 中断请求寄存器;

② 中断屏蔽寄存器;

③ 中断排队及编码电路;

④ 数据缓冲寄存器;

⑤ 控制/状态逻辑。

【例 9.11】 某计算机采用向量中断,4 个设备 A、B、C 和 D 的中断向量地址分别为 F5H、F4H、F7H 和 F6H。CPU 响应中断的优先次序为 A→B→C→D,CPU 只有一根中断请求输入线($\overline{\text{INTR}}$)和一根中断响应输出线(INTA)。请设计该中断的排队判优及向量编码线路,并说明其工作原理。

解:CPU 只有一根 $\overline{\text{INTR}}$线和一根 INTA 线,且 4 个设备的中断响应次序为 A→B→C→D,因此排队判优可以用串行优先链来完成,响应中断后,通过数据总线的 $D_7 \sim D_0$ 送出中断向量。

中断的排队判优及向量编码线路如图 9-17 所示。其工作原理如下。

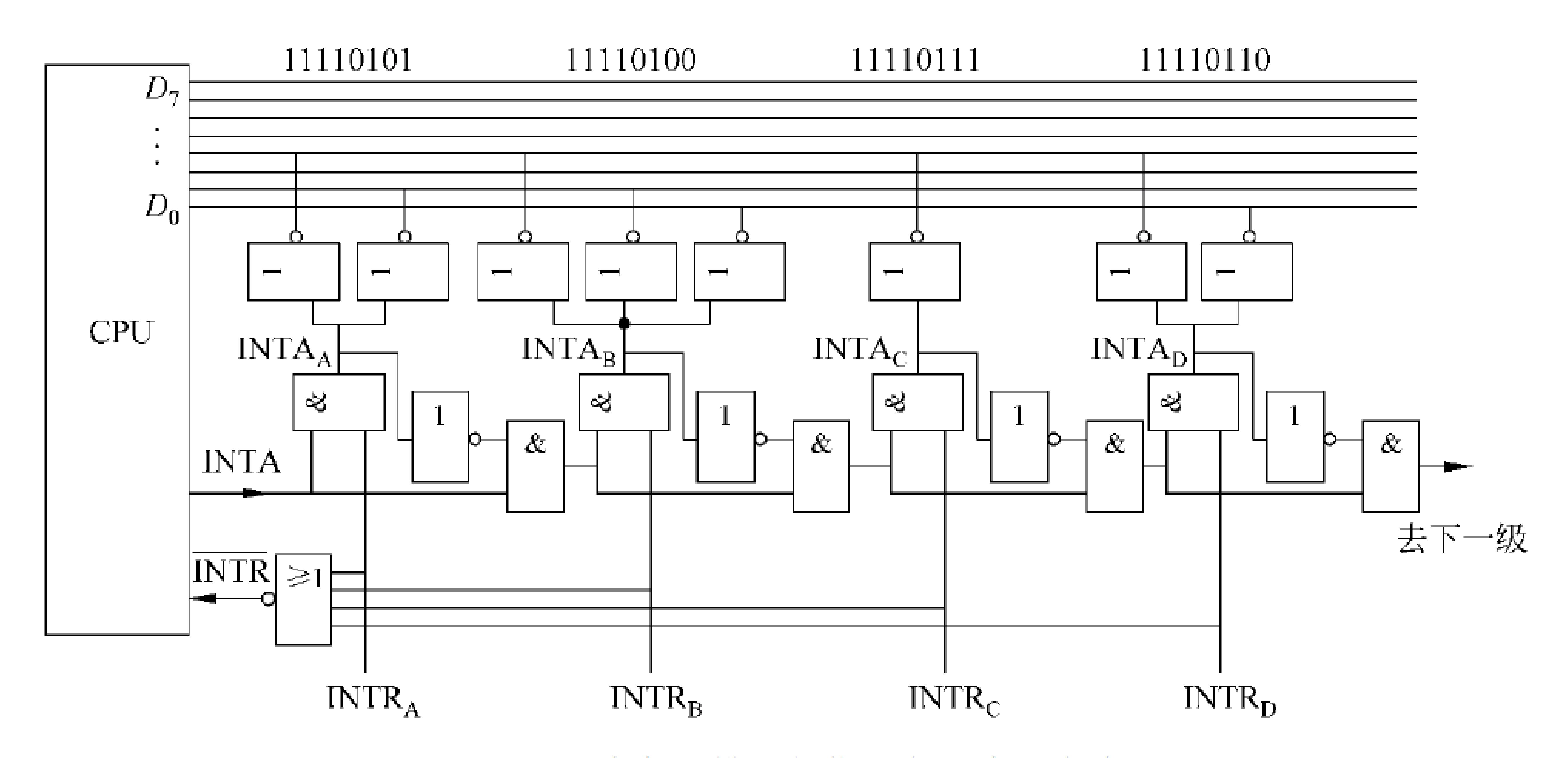


图 9-17 中断的排队判优及向量编码线路

所有设备的中断请求 $INTR_A$、$INTR_B$、$INTR_C$ 和 $INTR_D$,通过一根中断请求线 $\overline{\text{INTR}}$

送给 CPU。在开中断的情况下，CPU 在当前指令执行结束时，响应中断请求，发出 INTA 信号。INTA 信号串行的一次连接所有的中断源，若某设备没有中断请求，该设备就将中断响应信号 INTA 传送给下一个设备；若某设备有中断请求，该设备就封锁 INTA 信号，不再往下传送，使 INTA 信号终止在该设备上，同时产生该设备的中断允许信号 $INTA_i$，然后由向量地址编码器传输该设备的中断向量地址，并通过数据总线送入 CPU，CPU 便从相应的中断向量单元获得中断服务程序的入口地址，从而转去执行相应的中断服务程序。

【例 9.12】 在主存接收从磁盘送来的一批信息时：

(1) 假定主存的周期为 1μs，若采用程序查询方式传送，试估算在磁盘上相邻两数据字间必须具有的最短允许时间间隔是多少？

(2) 若改为中断方式传送，这个时间又会怎样？是否还有更好的传送方式？

(3) 在采用更好的传送方式下，假设磁盘上两数据字间的间隔为 1μs，主存又要被 CPU 占有一半周期时间，试计算这种情况下主存周期最少应该是多少？

解：(1) 因为主存周期为 1μs，则一个指令的执行时间至少为 1μs。若采用程序查询方式传送，两次数据之间至少需要 5 条指令，所以磁盘上相邻两数据字之间具有的最短允许时间间隔至少是 5μs。

(2) 若改为中断方式传送，这个时间不会减少，因为中断传送需要许多辅助操作，例如保护、恢复现场，开、关中断等。比中断传送方式更好的传送方式是 DMA 方式，它适合于对高速设备进行成组数据传送。

(3) 在 DMA 传送方式下，假设磁盘上两数据字之间的间隔为 1μs，主存又要被 CPU 占有一半周期时间，可以采用存储器分时法，在这种情况下，主存周期应该为 1μs÷2=0.5μs。

【例 9.13】 假定磁盘传输数据以 32 位的字为单位，传输速率为 1MB/s。CPU 的时钟频率为 50MHz。

(1) 程序查询的输入输出方式，一个查询操作需要 100 个时钟周期，求 CPU 为 I/O 查询所花费的时间比率，假定进行足够的查询以避免数据丢失。

(2) 用中断方式进行控制，每次传输的开销(包括中断处理)为 100 个时钟周期。求 CPU 为传输磁盘数据花费的时间比率。

(3) 采用 DMA 控制进行输入输出操作，假定 DMA 的启动操作需要 1000 个时钟周期，DMA 完成时处理中断需要 500 个时钟周期，如果平均传输的数据长度为 4KB，试问：在磁盘工作时，忽略 DMA 申请使用总线的影响，处理器将用多少时间比率进行输入输出操作？

解：根据题意可知，每传送一个字需要 4μs，CPU 的时钟周期为 0.02μs。

(1) 程序查询的输入输出方式，一个查询操作需要 100 个时钟周期，而时钟周期=0.02μs，所以每个查询操作需要 2μs，CPU 为 I/O 查询所花费的时间比率为 $\frac{0.02\times100}{4}=\frac{1}{2}$。

(2) 用中断方式法进行控制，每次传输的开销(包括中断处理)为 100 个时钟周期，而时钟周期=0.02μs，所以每次传输的开销时间=100×0.02μs=2μs，传送一个字的时间为 4μs，CPU 为传输磁盘数据花费的时间比率为 $\frac{0.02\times100}{4}=\frac{1}{2}$。

(3) 采用 DMA 控制进行输入输出操作，平均传输的数据长度为 4KB，根据数据传输率，传送时间 4KB÷1MB/s=4ms。又由于 DMA 的启动操作需要 1000 个时钟周期，即

$1000\times0.02\mu s=20\mu s$;DMA完成时处理中断需要500个时钟周期,即$500\times0.02\mu s=10\mu s$。所以,在磁盘工作时CPU为进行输入输出操作花费的时间比率为$\frac{0.02\times1500}{4000}=\frac{30}{4000}=\frac{3}{400}$。

【例9.14】 设一磁盘盘面共有磁道200道,盘面总存储容量为1.6MB,磁盘旋转一周时间为25ms,每道有4个区,各区之间有一间隙,磁头通过每个间隙需要1.25ms。

(1) 问磁盘通道所需最大传输速率是多少?

(2) 假设有人为上述磁盘机设计了一个与主机之间的接口,磁盘读出数据串行送入一个移位寄存器,每当移满16位后,向处理机发出一个请求交换数据的信号。处理机响应请求信号并取走移位寄存器的内容后,磁盘机再串行送入下一个16位的字,如此继续工作。如果现在已知处理机在接到请求交换信号后最长响应时间是3μs,这样的接口能否正常工作?应该如何改进?

解:(1) 每个磁道的容量=盘面总存储容量÷磁道数=1.6MB÷200≈8KB。

读一道数据的时间=(25−1.25×4)ms=20ms。

磁盘的数据传输速率=8KB÷0.02s=400KB/s。

(2) 因为磁盘的数据传输速率为400KB/s,所以磁盘准备一个16位字的时间为5μs,直接从移位寄存器送回数据的方案不能正常工作。因为移位寄存器保存一个字的时间仅为5μs÷16≈0.3μs,而响应时间可能达3μs,所以有可能失去数据。

改进方法:再设置一个发送寄存器,每当移位寄存器内满一个字时,就将其内容送发送寄存器保存,由发送寄存器发送数据。这个寄存器保存一个字的最短时间为5μs。

【例9.15】 某计算机系统字长为32位,包含两个选择通道和一个多路通道,每个选择通道上连接了两台磁盘机和两台磁带机,多路通道上连接了两台行式打印机、两台读卡机、10台终端。假定各设备的传输率如下:

磁盘机:800KB/s。

磁带机:200KB/s。

行打机:6.6KB/s。

读卡机:1.2KB/s。

终端:1KB/s。

计算该计算机系统最大I/O数据传输率。

解:为了保证通道不丢失数据,各种通道实际流量应该不大于通道的最大流量。本系统由3个不同的通道组成,系统的最大数据传输率等于所有通道最大通道传输率之和。

由于两个选择通道所连接的设备相同,只要计算其中一个通道的通道传输率即可。因为磁盘机的传输率大于磁带机,所以此类型通道的通道传输率为:

选择通道传输率=max{800KB/s,200KB/s}=800KB/s。

多路通道上的设备传输速率都比较低,则将它们组织成字节多路通道形式,那么该通道的最大传输率是通道上所有设备的数据传输率之和。即:

字节多路通道传输率=(6.6×2+1.2×2+1×10)KB/s=25.6KB/s。

计算机系统最大I/O数据传输率=2×选择通道传输率+字节多路通道传输率=(800×2+25.6)KB/s=1625.6KB/s。

*【例9.16】 下列选项中,能引起外部中断的事件是________。

A. 键盘输入　　B. 除数为0　　C. 浮点运算下溢　　D. 访存缺页

解：A。

分析：在这4个选项中，除键盘输入以外，其余3个选项都不是外部事件引起的中断。选项B、C的中断源是运算器，选项D的中断源是存储器。

*【例9.17】 单级中断系统中，中断服务程序内的执行顺序是________。

Ⅰ. 保护现场；Ⅱ. 开中断；Ⅲ. 关中断；Ⅳ. 保存断点；Ⅴ. 中断事件处理；Ⅵ. 恢复现场；Ⅶ. 中断返回

A. Ⅰ→Ⅴ→Ⅵ→Ⅱ→Ⅶ　　B. Ⅲ→Ⅰ→Ⅴ→Ⅶ

C. Ⅲ→Ⅳ→Ⅴ→Ⅵ→Ⅶ　　D. Ⅳ→Ⅰ→Ⅴ→Ⅵ→Ⅶ

解：A。

分析：程序中断有单级中断和多级中断之分，单重中断在CPU执行中断服务程序的过程中不能被再打断，即不允许中断嵌套。保存断点与关中断的任务是由硬件(中断隐指令)完成的，所以在单级中断系统中，中断服务程序内应完成的任务有：①保存现场；②中断事件处理；③恢复现场；④开中断；⑤中断返回。

*【例9.18】 某计算机有5级中断$L_4 \sim L_0$，中断屏蔽字为$M_4M_3M_2M_1M_0$，$M_i=1$ $(0 \leqslant i \leqslant 4)$表示对$L_i$级中断进行屏蔽。若中断响应优先级从高到低的顺序是$L_0 \to L_1 \to L_2 \to L_3 \to L_4$，且要求中断处理优先级从高到低的顺序为$L_4 \to L_0 \to L_2 \to L_1 \to L_3$，则$L_1$的中断处理程序中设置的中断屏蔽字是________。

A. 11110　　B. 01101　　C. 00011　　D. 01010

解：D。

分析：由于L_1的中断处理优先级下降，屏蔽字中需要3个0，所以可以将选项A、B排除掉。L_1需要对L_4、L_0、L_2开放，所以对应位应该为"0"。

*【例9.19】 某计算机处理器主频为50MHz，采用定时查询方式控制设备A的I/O，查询程序运行一次所用的时钟周期数至少为500。在设备A工作期间，为保证数据不丢失，每秒需对其查询至少200次，则CPU用于设备A的I/O的时间占整个CPU时间的百分比至少是________。

A. 0.02%　　B. 0.05%　　C. 0.20%　　D. 0.50%

解：C。

分析：对于设备A，每秒中查询至少200次，每次查询至少500个时钟周期，总的时钟周期数为100000。所以CPU用于设备A的I/O的时间占整个CPU时间的百分比至少为0.20%。

*【例9.20】 下列选项中，在I/O总线的数据线上传输的信息包括________。

Ⅰ. I/O接口中的命令字　　Ⅱ. I/O接口中的状态字　　Ⅲ. 中断类型号

A. 仅Ⅰ、Ⅱ　　B. 仅Ⅰ、Ⅲ　　C. 仅Ⅱ、Ⅲ　　D. Ⅰ、Ⅱ、Ⅲ

解：D。

分析：在I/O总线的数据线上传输的信息包括I/O接口中的命令字、状态字以及真正的数据，中断类型号也是通过数据线传输的。

*【例9.21】 响应外部中断的过程中，中断隐指令完成的操作，除保护断点外，还包括________。

Ⅰ. 关中断　　　　Ⅱ. 保存通用寄存器的内容

Ⅲ. 形成中断服务程序入口地址并送 PC

A. 仅Ⅰ、Ⅱ　　　B. 仅Ⅰ、Ⅲ　　　C. 仅Ⅱ、Ⅲ　　　D. Ⅰ、Ⅱ、Ⅲ

解：B。

分析：中断隐指令完成的操作有3个：①保存断点；②关中断；③引出中断服务程序(形成中断服务程序入口地址并送PC)。而保存通用寄存器内容的操作是由软件来实现，不是由中断隐指令实现的。

*【**例9.22**】 下列关于中断I/O方式和DMA方式比较的叙述中，错误的是________。

A. 中断I/O方式请求的是CPU处理时间，DMA方式请求的是总线使用权

B. 中断响应发生在一条指令执行结束后，DMA响应发生在一个总线事物完成后

C. 中断I/O方式下数据传送通过软件完成，DMA方式下数据传送由硬件完成

D. 中断I/O方式适用于所有外部设备，DMA方式仅适用于快速外部设备

解：D。

分析：中断I/O方式只适合于中、低速外部设备，而不是适合于所有的设备。

*【**例9.23**】 下列有关I/O接口的叙述中，错误的是________。

A. 状态端口和控制端口可以合用一个寄存器

B. I/O接口中CPU可访问的寄存器称为I/O端口

C. 采用独立编址方式时，I/O端口地址和主存地址可能相同

D. 采用统一编址方式时，CPU不能用访存指令访问I/O端口

解：D。

分析：采用统一编址方式时，没有专门的I/O指令，CPU将用访存指令来访问I/O端口。

*【**例9.24**】 若某设备中断请求的响应和处理时间为100ns，每400ns发出一次中断请求，中断响应所允许的最长延迟时间为50ns，则在该设备持续工作过程中，CPU用于该设备的I/O时间占整个CPU时间的百分比至少是________。

A. 12.5%　　　B. 25%　　　C. 37.5%　　　D. 50%

解：B。

分析：某设备每400ns发出一次中断请求，CPU用于中断请求的响应和处理时间为100ns，CPU用于该外设I/O的时间占整个CPU时间的百分比$=\frac{100}{400}\times100\%=0.25\times100\%=25\%$。

*【**例9.25**】 某计算机的CPU主频为500MHz，CPI为5(即执行每条指令平均需要5个时钟周期)。假定某外设的数据传输率为0.5MB/s，采用中断方式与主机进行数据传送，以32位为传输单位，对应的中断服务程序包含18条指令，中断服务的其他开销相当于2条指令的执行时间。请回答下列问题，要求给出计算过程。

(1) 在中断方式下，CPU用于该外设I/O的时间占整个CPU时间的百分比是多少？

(2) 当该外设的数据传输率达到5MB/s时，改用DMA方式传送数据。假定每次DMA传送块大小为5000B，且DMA预处理和后处理的总开销为500个时钟周期，则CPU用于该外设I/O时间占整个CPU时间的百分比是多少？(假设DMA与CPU之间没有访

存冲突)

解:(1) 已知主频为500MHz,则时钟周期=1÷500MHz=2ns,因为CPI=5,所以每条指令平均执行时间=5×2ns=10ns。

又已知每中断一次传送32位(4个字节),数据传输率为0.5MB/s,所以传送时间=4÷0.5MB/s≈8μs。

CPU用于该外设I/O共需20条指令(中断服务程序包括18条指令+其他开销折合2条指令),花费时间=20×10ns=200ns。

所以CPU用于该外设I/O的时间占整个CPU时间的百分比 $=\frac{200}{8000}\times 100\%=0.025\times 100\%=2.5\%$。

(2) 改用DMA方式传送数据,数据传输率为5MB/s,传送5000B的时间=5000÷5MB/s=1ms。

预处理和后处理的总开销时间=500×2ns=1μs。

CPU用于该外设I/O时间占整个CPU时间的百分比=预处理和后处理的总开销时间÷传送数据的时间 $=\frac{1}{1000}\times 100\%=0.001\times 100\%=0.1\%$。

9.4 同步测试习题及解答

9.4.1 同步测试习题

一、填空题

1. I/O接口按数据传送的宽度可以分为________和________两类。

2. CPU响应中断时需要保存当前现场,这里现场指的是________和________的内容,它们被保存到________中。

3. 在中断服务程序中,保护和恢复现场之前需要________中断。

4. DMA只负责在________总线上进行数据传送,在DMA写操作中,数据从________传送到________。

二、选择题

1. 将外围设备与主存统一编址,一般是指________。

A. 每台设备占一个地址码　　B. 每个外围接口占一个地址码
C. 接口中的有关寄存器各占一个地址码　　D. 每台外设由一个主存单元管理

2. 主机与设备传送数据时,采用________,主机与设备是串行工作的。

A. 程序查询方式　　B. 中断方式
C. DMA方式　　D. 通道方式

3. 当有中断源发出请求时,CPU可执行相应的中断服务程序。提出中断请求的可以是________。

A. 通用寄存器　　B. 专用寄存器
C. 外部事件　　D. Cache

4. CPU响应中断的时间是________。

A. 一条指令结束　　B. 外设提出中断
C. 取指周期结束　　D. 任一机器周期结束

5. 隐指令是指________。
A. 操作数隐含在操作码中的指令
B. 在一个机器周期里完成全部操作的指令
C. 隐含地址码的指令
D. 指令系统中没有的指令

6. 在中断周期,CPU 主要完成以下工作:________。
A. 关中断,保护断点,发中断响应信号并形成中断服务程序入口地址
B. 开中断,保护断点,发中断响应信号并形成中断服务程序入口地址
C. 关中断,执行中断服务程序
D. 开中断,执行中断服务程序

7. 向量中断是________。
A. 外设提出中断
B. 由硬件形成中断服务程序入口地址
C. 由硬件形成向量地址,再由向量地址找到中断服务程序入口地址
D. 以上都不对

8. 中断允许触发器用于________。
A. 向 CPU 发中断请求　　B. 指示正有中断在进行
C. 开放或关闭中断系统　　D. 指示中断处理结束

9. 中断屏蔽码的作用是________。
A. 暂停外设对主机的访问　　B. 暂停对某些中断的处理
C. 暂停对一切中断的处理　　D. 暂停 CPU 对主存的访问

10. 以下论述正确的是________。
A. CPU 响应中断期间仍执行原程序
B. 在中断过程中,若又有中断源提出中断请求,CPU 立即响应
C. 在中断响应中,保护断点、保护现场应由用户编程完成
D. 在中断响应中,保护断点是由中断隐指令自动完成的

11. DMA 方式是在________之间建立一条直接数据通路。
A. I/O 设备和主存　　B. 两个 I/O 设备
C. I/O 设备和 CPU　　D. CPU 和主存

12. 在 DMA 传送方式中,由________发出 DMA 请求。
A. 外部设备　　B. DMA 控制器
C. CPU　　D. 主存

13. 在 DMA 方式中,周期"窃取"是窃取一个________。
A. 存取周期　　B. 指令周期　　C. CPU 周期　　D. 时钟周期

14. 在采用 DMA 方式高速传输数据时,数据传送是________。
A. 在总线控制器发出的控制信号控制下完成的
B. 在 DMA 控制器本身发出的控制信号控制下完成的

C. 由 CPU 执行的程序完成的

D. 由 CPU 响应硬中断处理完成的

15. DMA 方式的接口电路中有程序中断部件，其作用是________。

A. 实现数据传送　　B. 向 CPU 提出总线使用权

C. 向 CPU 提出传输结束　　D. 发中断请求

16. DMA 方式________。

A. 既然能用于高速外围设备的信息传送，也就能代替中断方式

B. 不能取代中断方式

C. 也能向 CPU 请求中断处理数据传送

D. 内无中断机制

17. 通道程序是由________组成。

A. I/O 指令　　B. 通道控制字（或称通道指令）

C. 通道状态字　　D. 通道地址字

18. 对于低速输入输出设备，应当选用的通道是________。

A. 数组多路通道　　B. 字节多路通道

C. 选择通道　　D. DMA 专用通道

19. 一个计算机系统有 I/O 通道：①字节多路通道，带有传输速率为 1.2KB/s 的 CRT 终端 5 台，传输速率为 7.5KB/s 的打印机 2 台；②选择通道，带有传输速率为 1000KB/s 的光盘一台，同时带有传输速率为 800KB/s 的温盘一台；③数组多路通道，带传输速率为 800KB/s 及 600KB/s 的磁盘各一台，则通道的最大传输速率为________ KB/s。

A. 1821　　B. 2421　　C. 2621　　D. 3221

三、判断题

1. 一个外设接口中至少包含两个或两个以上的端口。（　　）

2. 输入输出接口中的数据端口是一个缓冲寄存器。（　　）

3. I/O 接口电路也是一种输入输出设备。（　　）

4. 在 I/O 接口电路中，主机和接口一侧的数据传送总是并行的。（　　）

5. 在允许多重中断的计算机系统中，只要外部有新的中断请求，就要打断正在处理的中断服务程序。（　　）

6. 中断请求的响应时间，必须安排在每个指令周期的末尾。（　　）

7. DMA 请求的响应时间，必须安排在每个指令周期的末尾。（　　）

8. 通道是实现外设和主存之间直接交换数据的控制器。（　　）

四、简答题

1. 简单叙述在中断系统中允许中断触发器的功能。

2. 在输入输出系统中，DMA 方式是否可以替代中断方式？

3. 试比较 I/O 通道控制方式和程序中断方式的特点。

4. 通道程序从哪里来？存放在哪里？

5. 在向量方式的中断系统中，为什么外设将中断向量放在数据总线上，而不放在地址总线上？

6. 试从下面 7 个方面比较程序查询、程序中断和 DMA 3 种方式的综合性能。

(1) 传送数据依赖软件还是硬件;

(2) 传送数据的基本单位;

(3) 并行性;

(4) 主动性;

(5) 传输速度;

(6) 经济性;

(7) 应用对象。

五、综合题

1. 某机中断分为8级(0~7),0级最高,7级最低,顺序排列。当某一个用户程序运行时,依次发生了3级、2级和1级中断请求,程序运行的轨迹如图9-18所示。如果用户程序在此3个中断请求发生前,用改变屏蔽字的方式将优先级改为0、5、3、4、1、2、6、7(从高到低),在上述中断请求情况下(中断请求产生时间严格按照上述顺序改变),请画出程序运行轨迹。

2. 某计算机的外部设备具有3级中断功能,中断响应次序基本上由硬件排队电路决定,但是可以利用各个外部设备控制器的中断屏蔽控制位来封锁本设备的中断请求信号。设所有中断服务程序的执行时间相同,均为T,在$5T$时间内共发生5次中断请求信号,如图9-19所示。其中,①表示1级中断设备发出的中断请求,其余类推。①的级别最高,②次之,③最低。

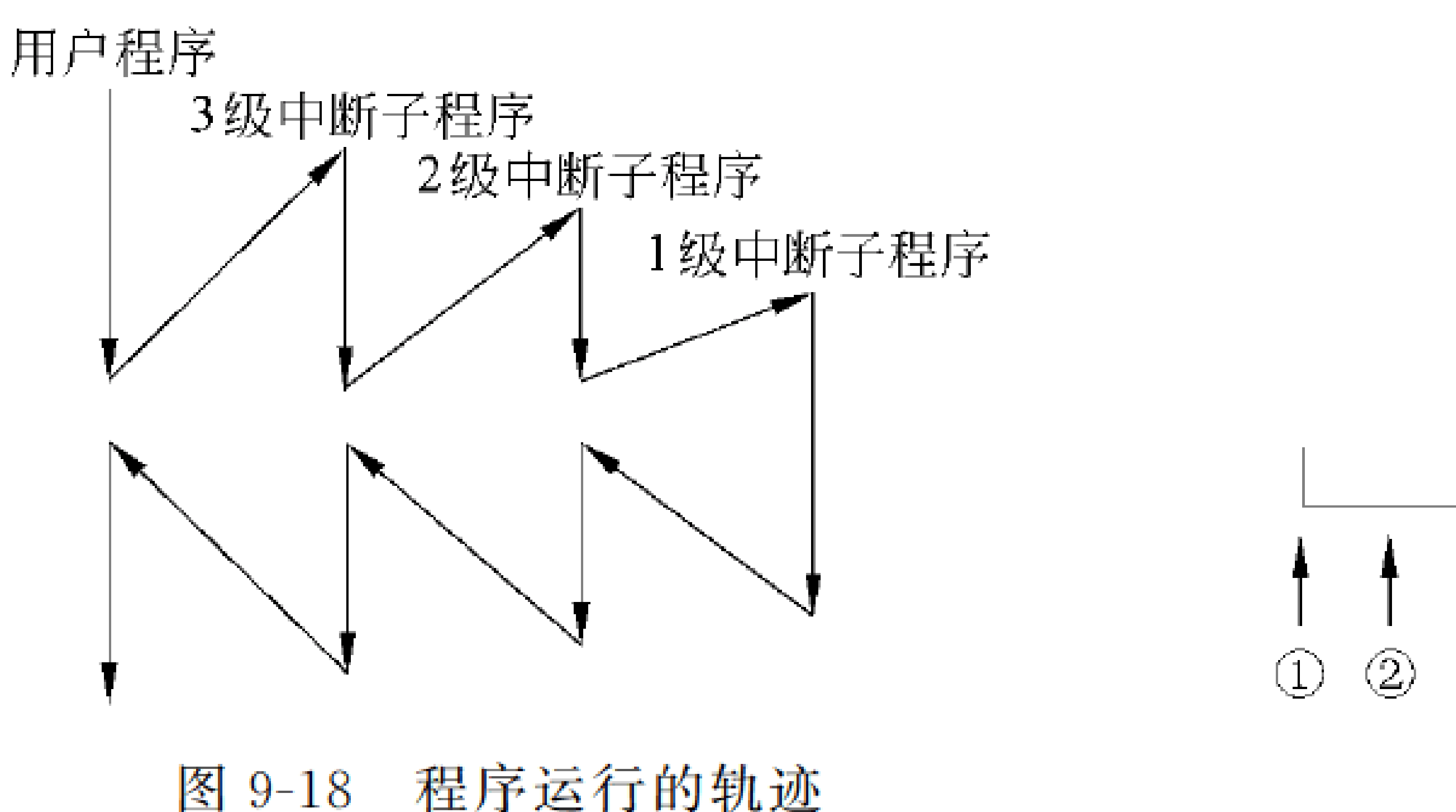


图9-18 程序运行的轨迹

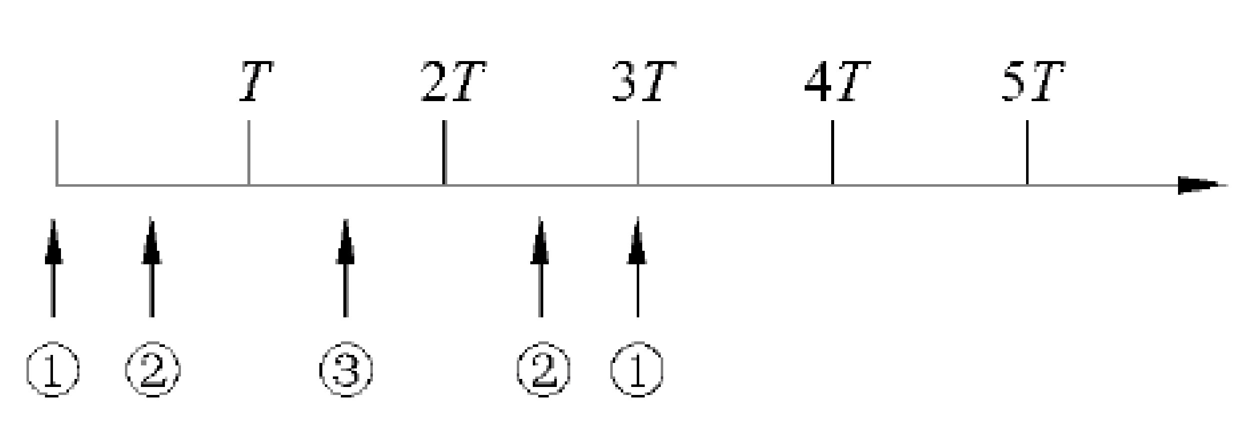


图9-19 中断出现顺序

(1) 请图示各个中断服务程序占用的时间段以及中断程序完成的次序。

(2) 软件进行干预,当执行中断服务程序③时,屏蔽②级中断。请图示各个中断服务程序占用的时间段及中断服务程序完成的次序。

3. 设某机有4个中断源A、B、C、D,其硬件排队优先次序为A>B>C>D,现要求将中断处理次序改为D>A>C>B。

(1) 写出每个中断源对应的屏蔽字。

(2) 按图9-20时间轴给出的4个中断源的请求时刻,画出CPU执行程序的轨迹。设每个中断源的中断服务程序时间均为20μs。

4. 假设某外设向CPU传送信息的最高速率为40K次/s,而相应中断服务程序的执行时间为40μs,问该外设是否可以采用程序中断方式?为什么?

5. 某中断系统响应中断需要50ns,总线中断服务程序至少需要150ns,其中60ns用于软件的额外开销。那么,该系统最大的中断频率是多少?中断额外开销时间占中断时间的

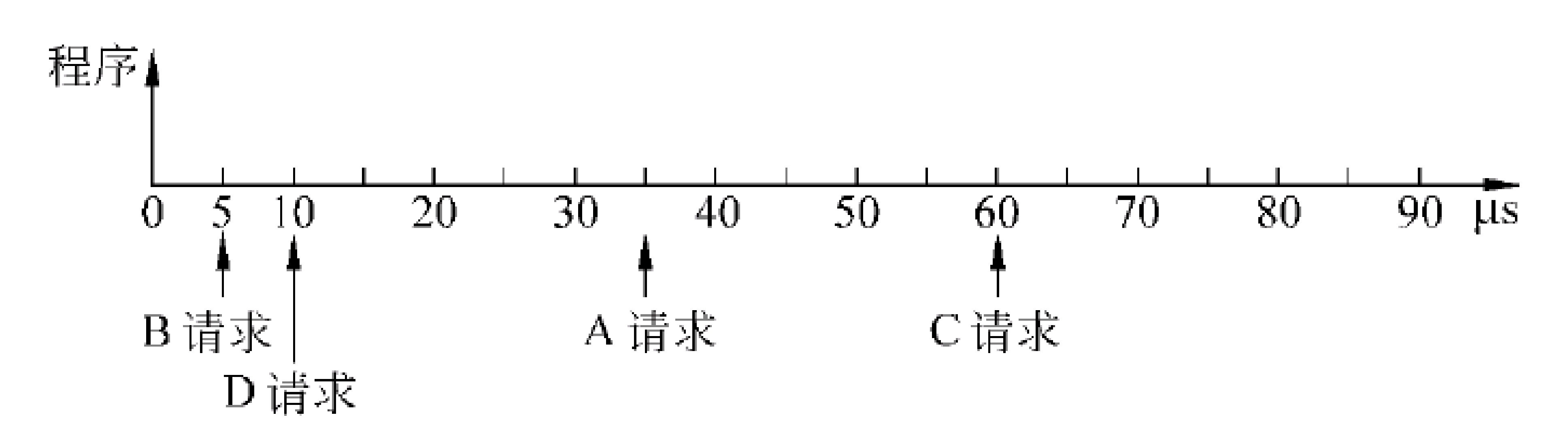


图 9-20　中断请求时刻

比例是多少？有一个字节设备，数据传输率为10MB/s，如果以中断方式且每次中断传送一个数据，那么该系统能实现这个传输要求吗？

6. 磁盘机采用DMA方式与主存通信，若主存周期为1μs，磁盘机的最高传输速率为多少才能适合存储器的要求？此时CPU必须处于什么状态？

7. 一个DMA接口可以采用周期窃取方式把字符传送到存储器，它支持的最大批量为400个字节。若存取周期为100ns，每处理一次中断需要5μs，现有的字符设备的传输率为9600bps。假设字符之间的传输是无间隙的，若忽略预处理所需要的时间，试问采用DMA方式每秒因数据传输需要占用处理器多少时间？如果完全采用中断方式，又需要占用处理器多少时间？

8. 今有一磁盘存储器，转速为3000r/min，分8个扇区，每扇区存储1KB。主存与磁盘传送数据的宽度为16b(即每次传送16位)。

(1) 描述从磁盘处于静止状态起将主存缓冲区中2KB传送到磁盘的整个工作过程。

(2) 假如一条指令最长执行时间为30μs，是否可采用在指令结束时响应DMA请求的方案？假如不行，应该采用怎样的方案？

9. 若输入输出系统采用字节多路通道方式，共有8个子通道，各子通道每次传送一个字节，已知整个通道最大传输速率为1200B/s，问每个子通道的最大传输速率是多少？若是数组多路通道，则每个子通道的最大传输速率又是多少？

9.4.2　同步测试习题解答

一、填空题

1. 串行接口，并行接口。

2. 断点状态，有关寄存器的内容，堆栈。

3. 关。在中断服务程序的保护和恢复现场之前要关中断，使处理现场工作不至于被打断。在中断服务程序的保护和恢复现场之后要开中断，为能再次响应中断请求做准备。

4. 系统，主存，外设。

二、选择题

1. C。统一编址时把I/O接口中的端口(有关寄存器)作为主存单元一样进行访问，通常每个端口占一个主存单元地址。

2. A。程序查询方式是由CPU执行一段输入输出程序来实现主机与外设之间数据传送的，所以主机和设备串行工作。

3. C。中断请求可以来自CPU外部也可以来自CPU内部。

4. A。CPU 响应中断的时间只能发生在每条指令执行完毕时。这是因为中断处理过程是程序切换过程,只有当一个程序的某条指令执行完毕才能切换到其他程序中。

5. D。中断隐指令并不是指令系统中的一条真正的指令,它没有操作码,所以中断隐指令是一种不允许,也不可能为用户使用的特殊指令。

6. A。在中断周期 CPU 执行中断隐指令,完成保存断点、关中断、形成中断服务程序入口地址 3 项操作。

7. C。向量中断通过硬件方式确定中断源,产生对应于中断源的向量地址,可以快速直接转向对应的中断服务程序。

8. C。中断允许触发器的作用是控制是否允许中断。当中断允许触发器=0 时,中断关闭(关中断),所有中断源的中断请求都不能得到响应;当中断允许触发器=1 时,中断允许(开中断),来自中断源的中断请求可以得到响应。

9. B。中断屏蔽码的作用是暂时剥夺部分中断源向 CPU 发出中断请求,利用中断屏蔽码,可以在不改变中断响应次序的情况下,改变中断处理的次序。

10. D。保存断点的操作是在中断周期由中断隐指令自动完成的。

11. A。直接存储器访问 DMA 方式是在外设和主存之间开辟一条"直接数据通道",在不需要 CPU 干预,也不需要软件介入的情况下,在两者之间进行的高速数据传送方式。

12. A。在 DMA 传送方式中,首先由外设向 DMA 控制器发出 DMA 请求信号,然后再由 DMA 控制器向 CPU 发出总线请求信号。

13. A。每次窃取一个存储周期进行一次数据传送,传送一个字节或一个字。

14. B。DMA 方式的数据传送过程不是由 CPU 执行程序完成的,而是在 DMA 控制器本身发出的控制信号控制下完成的。

15. C。DMA 控制器中的中断机构,用于数据块传送完毕时,向 CPU 提出中断请求,CPU 将进行 DMA 传送的结尾处理。

16. B。DMA 方式不能取代程序中断方式,如 DMA 的结束处理要通过中断来完成。

17. B。通道程序由通道指令组成。

18. B。字节多路通道是一种简单的共享通道,用于连接与管理多台低速设备,以字节交叉方式传送信息。

19. A。通道的最大传输速率=(1.2×5+7.5×2+1000+800)KB/s=1821KB/s。

三、判断题

1. √。

2. √。

3. ×。I/O 接口电路不是输入输出设备。

4. √。

5. ×。只有当新的中断请求的优先级别高于正在执行的中断服务程序时,才能打断正在处理的中断服务程序。

6. √。

7. ×。DMA 请求的响应时间,可以安排在每个机器周期的末尾。

8. √。

四、简答题

1. 允许中断触发器提供开中断和关中断功能。如果关中断，则不响应外部中断请求，如果开中断，则可响应外部中断请求。

2. 不可以。因为DMA方式的结束处理必须有中断方式的介入。

3. (1) 程序中断方式通过暂时中止CPU现行程序，转去执行中断服务程序实现；I/O通道控制方式则通过通道程序实现。

(2) 程序中断方式的中断服务程序与CPU现行程序是串行工作的；I/O通道控制方式的通道程序与CPU现行程序是并行工作的。

(3) I/O通道是集中独立的硬件，可连接多台快、慢速外设；程序中断方式只适于慢速外设，且每个外设都有自己的中断接口和中断服务程序。

4. 在具有通道的计算机中，CPU在进行一个输入输出操作之前，首先准备好通道程序，然后安排好数据缓冲区，再向通道和设备发启动命令。CPU准备好的通道程序存放在主存中，由通道读取并执行。通道在得到CPU的通知后，从主存中读取通道程序，并执行这个通道程序，从而完成输入输出操作。

5. 地址总线是单向的，只能用于CPU向存储器和外设传输地址信息，而不能用于外设向CPU传输信息，所以外设向CPU传输中断向量只能通过数据总线。

6. 表9-8列出了程序查询、程序中断和DMA 3种方式的综合性能。

表9-8　程序查询、程序中断和DMA 3种方式的综合性能

性　　能	程序查询	程序中断	DMA
数据传送	依赖软件	依赖软件	依赖硬件
传送数据的基本单位	字	字	信息块
并行性	CPU与I/O串行	CPU与I/O并行传输与主程序串行	CPU与I/O并行传输与主程序并行
主动性	CPU	设备	设备
传输速度	慢	慢	快
经济性	费用低	介于程序和DMA之间	费用高
应用对象	低速	较低	高速成批传输

五、综合题

1. 改变屏蔽字后程序运行的轨迹如图9-21所示。

2. (1) 没有软件进行干预时，实际处理顺序为：①→②→③。

中断服务程序占用的时间段以及中断服务程序完成的次序如图9-22所示。

(2) 由于进行软件干预，实际处理的次序发生变化，虽然②的响应次序高于③，但是处理次序却低于③，当③先到来时，②并不能中断它。

此时，中断服务程序占用的时间段以及中断服务程序完成的次序如图9-23所示。

3. (1) 在中断处理次序改为D>A>C>B后，每个中断源新的屏蔽字如表9-9所示。

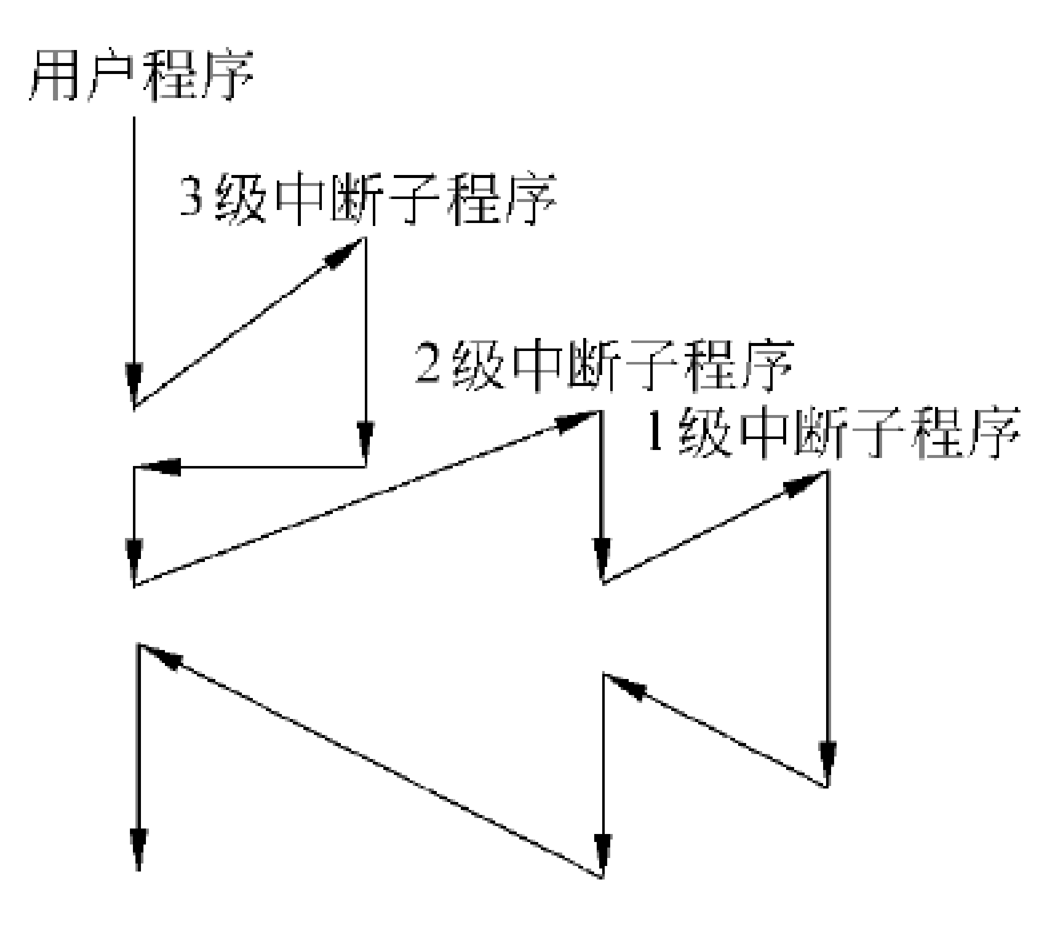


图 9-21 改变屏蔽字后程序运行的轨迹

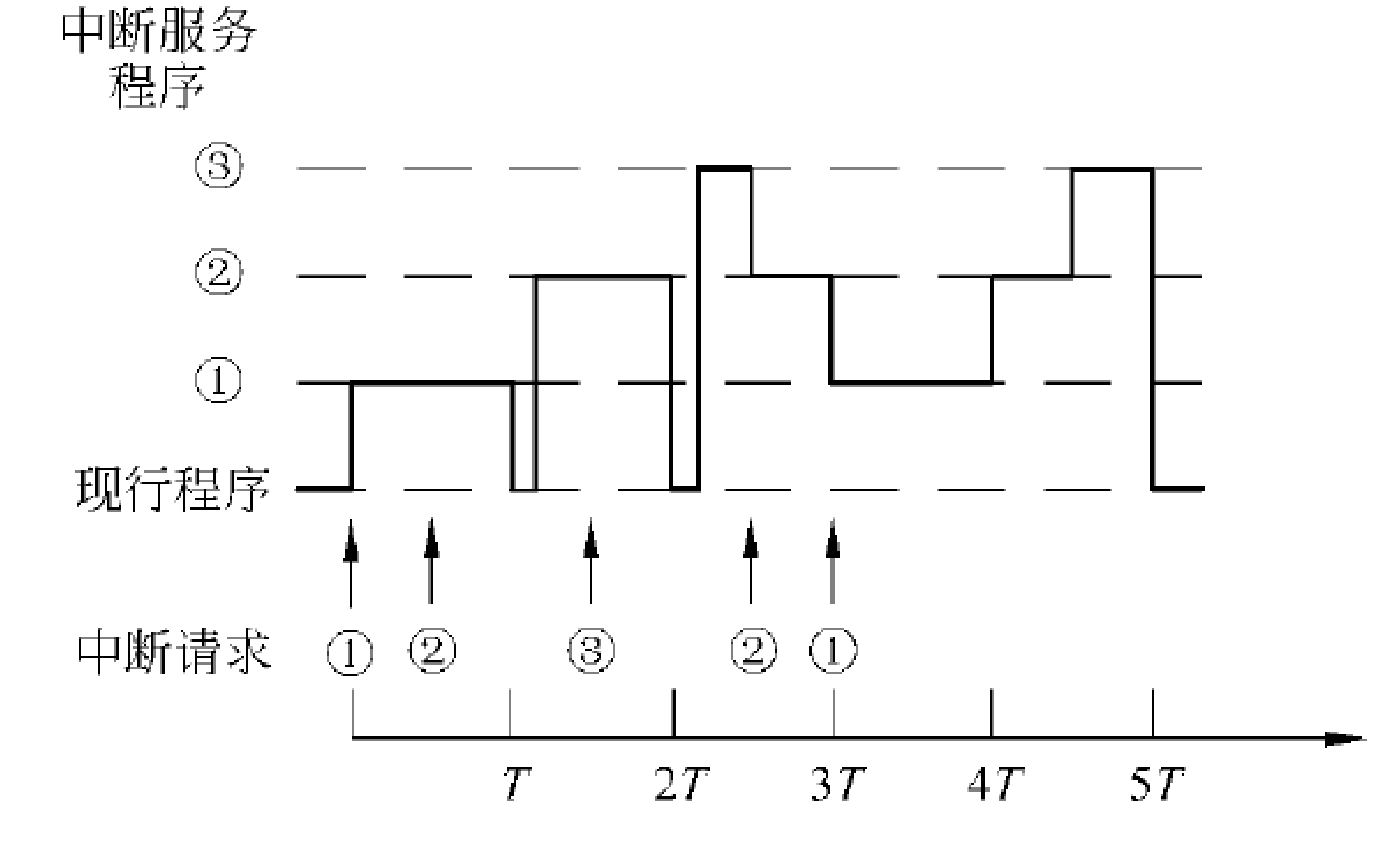


图 9-22 没有软件进行干预时中断示意图

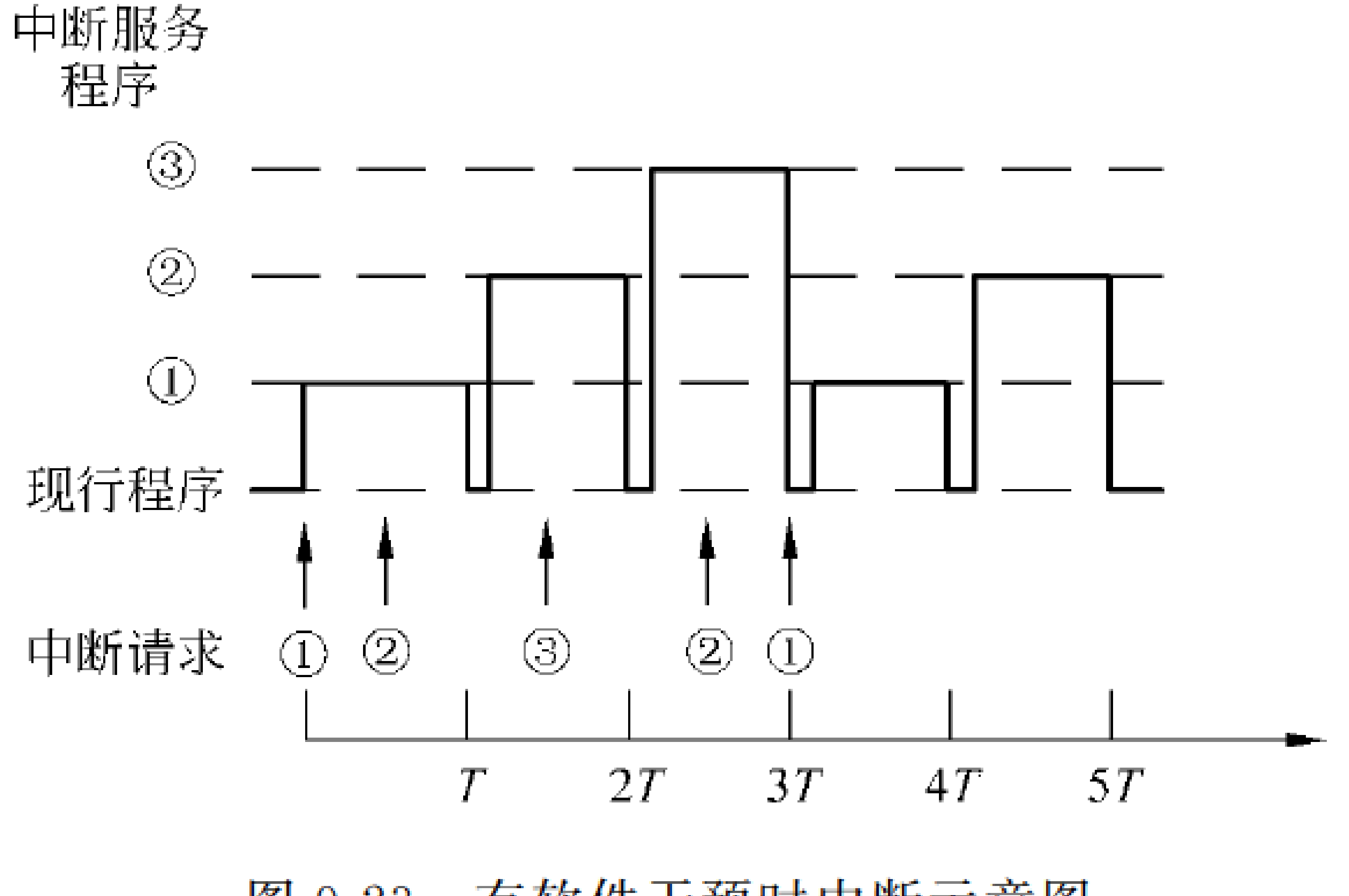


图 9-23 有软件干预时中断示意图

表 9-9 新的中断屏蔽码

中断源	中断屏蔽码			
	A	B	C	D
A	1	1	1	0
B	0	1	0	0
C	0	1	1	0
D	1	1	1	1

(2) 根据新的处理次序,CPU 执行程序的轨迹如图 9-24 所示。

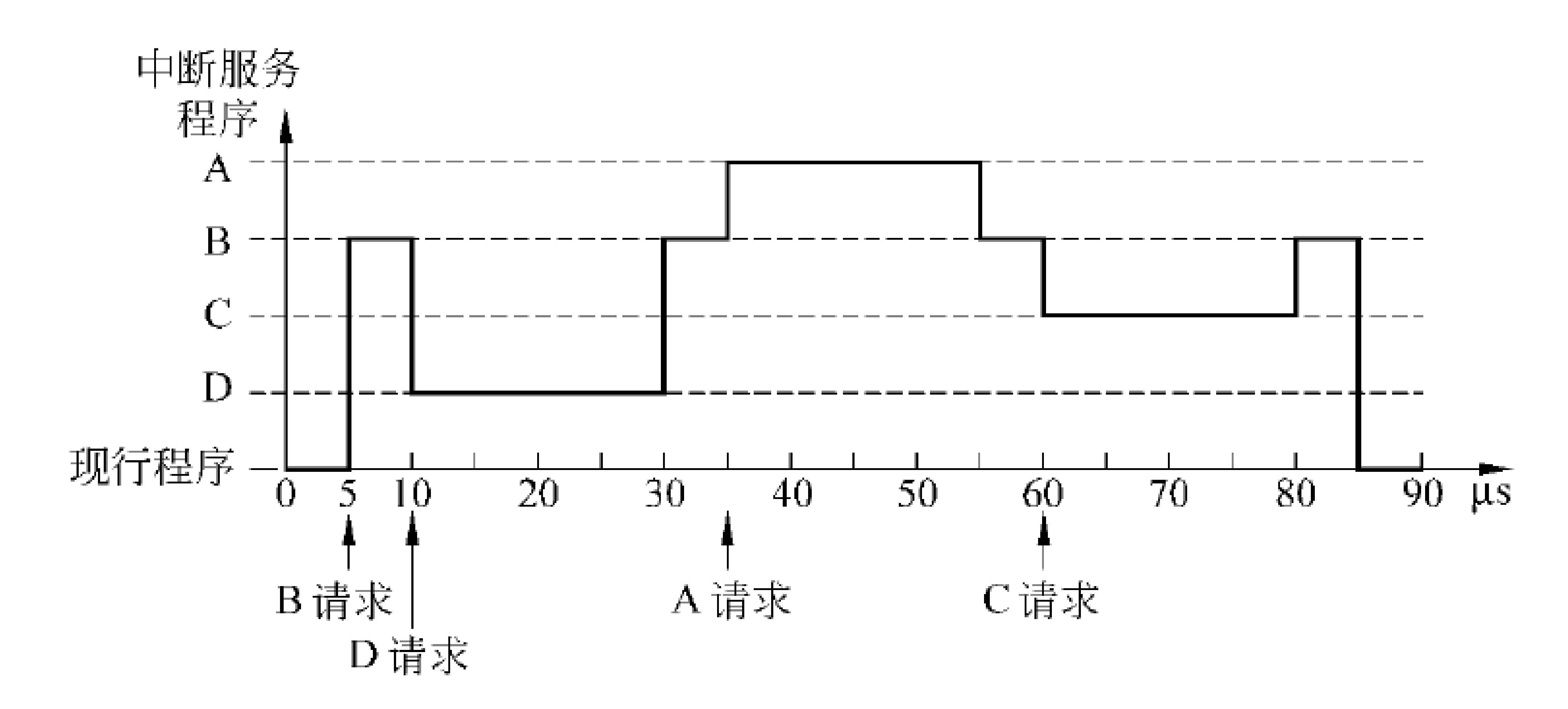


图 9-24 CPU 执行程序的轨迹

4. 外设传送一个数据的时间$=\frac{1}{40}\times10^{-3}s=25\mu$s,所以请求中断的周期为 25μs,而相应的中断服务程序的执行时间为 40μs,会丢失数据,所以不能采用程序中断方式。

5. 因为最短的中断间隔时间=最短的中断时间=50ns+150ns=200ns,所以最大的中断频率=1÷200ns=5×10^{6}Hz=5MHz。

因为中断额外开销时间=中断系统响应时间+软件额外开销=50ns+60ns=110ns,所以中断额外开销时间占中断时间的比例=110÷200=55%。

设备数据传输率为10MB/s，即传输数据的间隔时间=100ns<最短的中断间隔时间，所以该系统不能实现这个传输要求。

6．最高传输速率$=\frac{1}{1\mu s}=$1M次/s，此时CPU必须处于等待状态。

7．根据字符设备的传输率9600bps，则9600b/s÷8=1200B/s。

若采用DMA方式，传送1200个字符共需1200个存取周期，每传送400个字符需中断一次，因此DMA方式每秒因数据传输占用处理器的时间为$0.1\mu s\times1200+5\mu s\times(1200\div400)=120\mu s+15\mu s=135\mu s$。

若采用中断方式，每传送一个字符要申请一次中断请求，每秒因数据传输占用处理器的时间为$5\mu s\times1200=6000\mu s$。

8．(1) 主程序应先启动磁盘驱动器，并向接口发送设备地址、主存缓冲区首地址、传送字数(1KW=2KB)等预处理工作。磁盘寻道并等待转到访问的扇区后，通过接口发出1024个DMA请求，传送1KW个数据。当数据传送完后，接口向CPU发中断请求，由中断服务程序实现停止磁盘工作等后处理工作。

(2) 数据传输率$=8\text{KB}\times\frac{3000\text{转}}{60\text{s}}=400\text{KB/s}$，即每16位数据保持最短时间$=\frac{2}{400\text{KB/s}}=5\mu s$，而一条指令最长执行时间为$30\mu s$，所以如果指令结束时再响应DMA请求可能丢失数据，应该使每个机器周期结束时都可以响应DMA请求。

9．每个子通道的最大传输速率是1200B/s÷8=150B/s。

若是数组多路通道，则每个子通道的最大传输速率应为1200B/s。

参 考 文 献

[1] 蒋本珊. 计算机组成原理(第 3 版). 北京：清华大学出版社，2013

[2] 蒋本珊. 计算机组成原理教师用书(第 2 版). 北京：清华大学出版社，2009



普通高等教育"十一五"国家级规划教材
21世纪大学本科计算机专业系列教材

近期出版书目

- 计算概论(第2版)
- 计算概论——程序设计阅读题解
- 计算机导论(第3版)
- 计算机导论教学指导与习题解答
- 计算机伦理学
- 程序设计导引及在线实践
- 程序设计基础(第2版)
- 程序设计基础习题解析与实验指导
- 如何编写C程序
- 程序设计基础(C语言)(第2版)
- 程序设计基础(C语言)实验指导(第2版)
- 离散数学(第3版)
- 离散数学习题解答与学习指导(第3版)
- 数据结构(STL框架)
- 算法设计与分析
- 算法设计与分析习题解答与学习指导
- 算法设计与分析(第3版)
- 算法设计与分析习题解答(第3版)
- C++程序设计(第2版)
- Java程序设计
- 面向对象程序设计(第3版)
- 形式语言与自动机理论(第3版)
- 形式语言与自动机理论教学参考书(第3版)
- 数字电子技术基础
- 数字逻辑
- FPGA数字逻辑设计
- 计算机组成原理(第3版)
- 计算机组成原理教师用书(第3版)
- 计算机组成原理学习指导与习题解析(第3版)
- 微机原理与接口技术
- 微型计算机系统与接口(第2版)
- 计算机组成与系统结构
- 计算机组成与系统结构习题解答与教学指导
- 计算机组成与体系结构(第2版)
- 计算机系统结构教程
- 计算机系统结构学习指导与题解
- 计算机系统结构实践教程
- 计算机操作系统(第2版)
- 计算机操作系统学习指导与习题解答
- 编译原理
- 软件工程(第2版)
- 计算机图形学
- 计算机网络(第3版)
- 计算机网络教师用书(第3版)
- 计算机网络实验指导书(第3版)
- 计算机网络习题解析与同步练习
- 计算机网络软件编程指导书
- 人工智能
- 多媒体技术原理及应用(第2版)
- 计算机网络工程(第2版)
- 计算机网络工程实验教程
- 信息安全原理及应用